KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Algebraic Formalizations of Constructive Knowledge

Linde Vanbesien

Supervisors:
Prof. dr. Marc Denecker
(KU Leuven)
Prof. dr. Bart Bogaerts
(Vrije Universiteit Brussel and KU Leuven)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

October 2025

Algebraic Formalizations of Constructive Knowledge

Linde VANBESIEN

Examination committee:
Prof. Em. dr. ir. Lucie Vandewalle, chair
(KU Leuven)
Prof. dr. Marc Denecker, supervisor
(KU Leuven)
Prof. dr. Bart Bogaerts, supervisor
(Vrije Universiteit Brussel and KU Leuven)
Prof. dr. ir. Frank Piessens (KU Leuven)
Prof. dr. Joost Vennekens (KU Leuven and Vrije Universiteit Brussel)
Prof. dr. dr. Geraint Wiggins (Vrije Universiteit Brussel and Queen Mary University of London)
Prof. dr. Bas Ketsman (Vrije Universiteit Brussel)

Prof. dr. Jesse Heyninck
(Open Universiteit)
Prof. dr. Angelos Charalambidis
(Harokopio University of Athens)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

October 2025

© 2025 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Linde Vanbesien, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Use of Generative AI I did not use generative AI assistance tools during the research/writing process of my thesis, except for mere language assistance. The text/code/images in this thesis are my own (unless otherwise specified) and generative AI has only been used in accordance with the KU Leuven guidelines and appropriate references have been added. I have reviewed and edited the content as needed and I take full responsibility for the content of the thesis.

Preface

Writing this dissertation marks the end of a long and inspiring journey through the fascinating world of constructive knowledge and its formal underpinnings. Over the past years, I have explored how the process of construction itself can serve as a common thread uniting different forms of reasoning and definition. This work ultimately led to a generalized version of Approximation Fixpoint Theory (AFT) and new insights into how constructive knowledge can be expressed and understood within knowledge representation.

This research was carried out at on the one hand the KRR-subgroup of the DTAI Research group within the Computer Science Department at the KU Leuven, and on the other hand at the AI lab within the Computer Science Department at the Vrije Universiteit Brussel (VUB). It was made possible thanks to the generous support of the FWO under project G0B2221N titled 'Approximation Fixpoint Theory as a General Algebraic Theory of Constructive Knowledge'. I am deeply grateful for their trust and for giving me the freedom and resources to pursue this research.

I owe my deepest gratitude to my supervisors, Prof. dr. Marc Denecker at KU Leuven and Prof. dr. Bart Bogaerts at VUB. Their guidance, insight, and seemingly inexhaustible source of knowledge have shaped this work in more ways than I can express. They challenged me to think deeper, encouraged me to explore ambitious ideas, and supported me through every high and low along the way.

The true luxury of doing a joint PhD lies in having two supervisors to rely on, each with their own strengths and perspectives. I would like to thank Marc for his endless stream of ideas and inspiration. His devotion to research and to the field is profoundly motivating and pushed me to dig deeper into my own potential. I am equally grateful to Bart for his fast, efficient, and constructive feedback, always delivered with remarkable ease. His ability to grasp complex and abstract concepts, such as those found in AFT, in such a short time never ceased to amaze me.

My supervisors complemented each other perfectly: while Marc paid close attention to detail and taught me to be precise and clear (although I still have a long way to go),

Bart ensured I never lost sight of the bigger picture, both in my research and beyond.

I would also like to sincerely thank the other members of my supervisory committee: Prof. dr. ir. Frank Piessens, Prof. dr. Joost Vennekens, and Prof. dr. dr. Geraint Wiggins. Their feedback throughout my PhD was invaluable and provided me with much-needed outside perspectives. They encouraged me to look beyond the boundaries of the field in which I was academically raised. Their suggestions and ideas never disappointed; each discussion improved my work and contributed to making me a more well-rounded researcher.

Moreover, I would like to thank the remaining members in my examination committee, i.e., Prof. dr. Bas Ketsman, Prof. dr. Jesse Heyninck and Prof. dr. Angelos Charlambidis, for taking the time to read my thesis with care and for their valuable feedback and questions. Their engagement with my work means a great deal to me. I am convinced I did not have the easiest thesis to read as the topic is fairly abstract in nature. Nevertheless they provided me amazing feedback that greatly improved my text and even my understanding of the topic, even in the later stages of my PhD journey. A special word of thanks also goes to the chair of my examination committee, i.e., Prof. Em. dr. ir. Lucie Vandewalle, whose comforting presence and excellent organisational skills ensured that my defences went smoothly. Her calm and supportive guidance helped avoid a more stressful experience.

I would also like to thank Prof. Em. dr. Gerda Janssens and Prof. Em. dr. Maurice Bruynooghe, who felt like supervisors from afar as more experienced members of the research group. They were always ready for discussion and generously shared their time and insights, creating many genuine moments of joy during our weekly KRR meetings.

A heartfelt thank you also goes to my (ex-)colleagues and friends at, among others, the KRR and AI lab research groups. The discussions, shared frustrations, and laughter made these years not only productive but genuinely enjoyable. This journey would not have been the same without your support and companionship. I began my PhD during the Covid period in a fully online setting. Nevertheless, my then closest colleagues, i.e., Matthias Vanderhallen, Ruben Lapauw, Simon Marynissen, Pierre Carbonelle and Đorđe Markovic, effortlessly welcomed me into the group. With invaluable tips and tricks, they helped set me on the path toward a successful completion of my PhD.

As all of them have since graduated, their places have been filled by newer members of the group. Yet, the group's dynamic and supportive atmosphere has remained constant. Every single colleague has been an amazing companion, giving me the space to be myself and contributing in some way to my life, both professionally and beyond.

In particular, I would like to thank my longest-standing colleague, Đorđe Markovic, whose PhD journey unfolded alongside mine. Being consistently at the same stage of our research, we shared countless discussions and heart-to-heart conversations

throughout our journey. I would also like to personally thank Robbe Van den Eede, who never failed to brighten my days with (sometimes terrible, sometimes less terrible) puns. Both of them shared an office with me and had to tolerate my talkative nature, yet they never complained. They served as a soundboard, occasionally as guides, and always, above all, as friends.

Similarly, a special mention is warranted for my office mates at the VUB office. I greatly appreciate Samuele Pollaci's outlook on life and his smart, kind and open nature. Whether it is for venting frustrations or for solving research-related questions, he is always ready to offer a helping hand and I am proud to call him a friend. I have found another pair of like-minded friends in my colleague Dieter Vandesande and his partner Astrid Vervoort. I want to especially thank Dieter for his equally big appreciation of coffee breaks as mine and his willingness to talk about anything and everything.

Next, I would like to thank my current closest colleagues: Carlos Cantero, for fearlessly asking questions that challenge my worldview and for always offering philosophical food for thought; Daimy Van Caudenberg, for being unapologetically herself and bringing a refreshing energy to the team; Max Moeller, for always bringing a smile and spreading joy; Alexander Ek, for his dry sense of humor and insightful perspective; Berhan Oumer Adame, for his comically dry appreciation of my enthusiasm and quirks; and Nils Froleyks, for sharing new experiences and knowledge that enrich our group.

A special thanks also goes to my other colleagues and friends with whom I've shared countless lunches, coffee breaks, and evenings filled with interesting philosophical and personal discussions, providing me with a wonderful social network alongside the work. In particular, I would like to give a shout-out to Vida Ranjbar, Nima Rahimi Foroushaani, Marton Bognar, Senne Berden, Weihong Wang, Héloïse, Pieter, Ignace Bossuyt, Denis Carnier and Elias Storme.

Finally, I am grateful to my family and friends outside of academia, for reminding me that there is life beyond research, and for believing in me even when I doubted myself.

A special appreciation goes to my international friends Shobhit Arora for being a constant positive and optimistic influence on my life, Vera Danilova for being the sweetest role model anyone can wish for, and Adrian Saldanha whose calm demeanor and true friendship have frequently warmed my heart. Together they have broadened my horizons, and have deepened my appreciation for international connections.

Ook wil ik graag de Vlaamse Zweefvliegclub Phoenix en haar leden bedanken. Ik heb hier menige dagen doorgebracht achter mijn computer, werkend aan mijn thesis, zowel in het clublokaal als op de vele vliegvelden, binnen- en buitenland. De club werd soms letterlijk mijn bureau, en enkele leden hebben mij op hun eigen manier ondersteund, bijvoorbeeld door me af en toe te trakteren op een ice-tea.

Ik wil graag een dansende Amélie persoonlijk bedanken voor de nodige inspiratie op

precies het juiste moment, waardoor ik eindelijk een doorbraak bereikte na maanden zoeken naar een bewijs. Ook wil ik Julien en Timothy bedanken voor hun hulp bij het meedenken over de voorbeelden in mijn thesis.

Een zeer speciale dank gaat uit naar Laure en Jarno voor hun steun op verschillende vlakken: van het organiseren van BBQ's na een lange werkdag, tot het beschikbaar stellen van hun tuin tijdens een elektriciteitsnoodgeval, en zelfs het meehelpen zoeken naar de juiste outfit voor mijn verdediging.

Mijn vriendinnen uit mijn middelbare schoolperiode, i.e., Ianthe, Leen, Sarah en Heleen, verdienen ook een welgemeende dankjewel. Ze stonden altijd klaar met een luisterend oor en zorgden voor opbeurende, motiverende avonden vol gelach. Zij hebben me voor altijd een voorbeeld gegeven van echte vriendschap, en dat zal al mijn andere vriendschappen kleuren.

Evenzo hebben de dinkies, Tom en Karolien, mij menig onvergetelijke avond bezorgd waarop ik mijn zorgen even kon vergeten en me gewoon kon focussen op het eten en het gezelschap. Van karaoke's tot triatlons: ik koester de vele herinneringen die we samen hebben gemaakt in de afgelopen jaren.

Ook wil ik graag Arno en zijn partner Emily bedanken. Onze afspraken deden altijd wonderen voor mijn zelfvertrouwen dankzij jullie motiverende woorden en complimenten.

Lotte, Linde, Joke, Robin, Toon en Annick uit mijn studententijd stonden eveneens altijd klaar met bemoedigende woorden en gelukwensen. Talloze herinneringen, reizen en frustratiebladeren later staan ze nog steeds voor me klaar om me te steunen en te vieren door dik en door dun.

Verder verdienen ook Achim en zijn partner Aliona een vermelding. Achim is al meer dan twaalf jaar een inspirerende aanwezigheid in mijn leven. Hij is buitengewoon intelligent en behulpzaam en heeft me doorheen mijn studies en doctoraat meerdere keren geholpen met raad en uitleg.

Daarnaast wil ik graag Thomas bedanken. Thomas is een vriend op wie je kan bouwen, altijd bereid om te helpen en te luisteren. Hij neemt graag een mentorrol op en heeft me vaak begeleid in mijn zoektocht naar mezelf. Hij heeft me gesteund doorheen mijn diepste dalen en hoogste pieken.

Ook wil ik graag Hanne en Maarten in de verf zetten. Als oud-kotgenoten kennen ze mij in al mijn glorie, de goede en de minder goede kanten. Toch aanvaarden ze me volledig, en ik heb me zelden zo gezien en geapprecieerd gevoeld als bij hen. Een extra speciale dankjewel aan Hanne voor haar onvoorwaardelijke steun en de vele pep-talks tijdens mijn doctoraat. Vaak voelde Hanne meer als een persoonlijke cheerleader dan iets anders.

Tot slot richt ik mij tot mijn familie. Aan de ene kant wil ik de familie De Smet en de familie De Glas bedanken. Ze hebben mij als aanhangsel volledig geaccepteerd, en ik ben trots dat ik hen tot mijn familie mag rekenen. Een speciale dankjewel aan Greet voor haar vele bemoedigende woorden, haar vertrouwen in mijn kunnen en haar alomtegenwoordige enthousiasme. Ook bedank ik Gilberte en Michel voor hun vele zorgen, van de was en strijk tot een appeltaartje of een glas 'champagne'. Cato en Arno dank ik graag voor de vele interessante gesprekken en aangename dagen, en Wolf voor zijn ongefilterde tekenen van affectie. Ook Jan, Kathleen en Birte verdienen een persoonlijke vermelding voor de rol die ze met enthousiasme in mijn leven hebben opgenomen.

Daarnaast wil ik ook de familie Vanbesien en de familie Desodt bedanken. Je familie kies je niet zelf, maar mocht ik kunnen kiezen, dan koos ik hen gewoon opnieuw. Een welgemeende dankjewel aan mijn ouders Anne en Frederik, wiens liefde, steun en opvoeding mij gebracht hebben tot waar ik vandaag sta. Woorden schieten te kort om te beschrijven hoeveel ze al voor mij gedaan hebben. Ook mijn broers Ewout (en partner Pamela), Rowald en Tiebe wil ik bedanken voor hun steun de afgelopen jaren, elk op hun eigen, typisch broederlijke manier, van geplaag tot complimenten. Mijn zus Marte (en partner Simon) verdient eveneens een speciale plek in dit dankwoord: weinig mensen kennen mij zo goed als zij, en ik weet dat ik bij haar altijd terechtkan voor een luisterend oor.

En dan kom ik bij de allerbelangrijkste persoon in mijn leven, i.e., mijn kip Pieter-Jan. Hij was de kalmte in mijn stormen en heeft me (bijna zonder klagen) door weer en wind gesteund. Met de nodige dosis aaitjes, of gewoon als kok en klankbord, stond hij altijd voor me klaar, ook wanneer de deadlines van mij een hoogst onaangename levenspartner maakten. Hij slaagde er telkens in om een glimlach op mijn gezicht te toveren. Zonder mijn kip was ik nooit zo ver geraakt.

Tot slot wil ik graag mijn oma herdenken. Het scheelde niet veel, maar helaas moet ze mijn afstuderen missen. Als laatste van mijn vier grootouders had ze een grote rol om op te vullen, maar haar trots was groot genoeg voor die van vier samen. Haar trots is een drijvende kracht geweest die een belangrijke rol heeft gespeeld bij het volbrengen van mijn doctoraat, een kracht die ik voor de rest van mijn leven met me meedraag.

Abstract

Constructive knowledge plays a central role in knowledge representation (KR). Many KR formalisms, ranging from logic programs and inductive definitions to auto-epistemic and non-monotonic logics, are essentially constructive in nature: they describe processes that construct entities step by step from basic rules.

This thesis develops a systematic theoretical understanding of constructive knowledge by focusing on the unifying element underlying its diverse forms: the construction process. The work pursues three main goals: (1) generalizing and unifying the treatment of construction processes, (2) clarifying and extending existing tools to formalize constructive knowledge, and (3) demonstrating the applications of these tools in knowledge representation using concrete examples.

We begin by analyzing monotonic constructive definitions, showing that despite their varied origins, they share a common nature: they correspond to a *semantic operator* acting on an ordered *construction space*, with the construction process formalized as a monotonic induction. Distinctions across domains arise primarily from the underlying construction space, and making these explicit enables the development of logics supporting diverse constructive definitions.

For non-monotonic operators, Approximation Fixpoint Theory (AFT) provides a principled framework to capture the intended semantics. On the one hand it formalizes the construction process by means of well-founded inductions. On the other hand, it offers a clear algorithm to compute the model using fixpoint-based semantics of approximating operators.

While AFT captures non-monotonic induction, its abstract formulation has limited uptake in the answer set programming and logic programming communities which have been argued to model induction. This thesis bridges that gap by reformulating AFT in terms familiar to these communities. Thus we demonstrate its applicability to concrete KR tasks, in particular to extending ASP with aggregates.

Building on this, we extend AFT itself by relaxing the notions of approximation

space and approximant. Our generalized framework (gAFT) gives rise to multiple instantiations based on the same fundamental principles. It preserves both the fixpoint-based semantics and constructive well-founded inductions, while accommodating a broader range of domains and operators. Concrete instantiations, including flower-based AFT (fAFT) and relaxed interval-based AFT, demonstrate the framework's applicability to concrete applications such as non-monotonic recursive definitions, auto-epistemic logic and logic programming.

Overall, this work unifies previously fragmented perspectives, clarifies and extends foundational tools, and bridges theory and practice in constructive knowledge representation.

Beknopte samenvatting

Constructieve kennis speelt een belangrijke rol in Kennis Representatie (KR). Veel KR formalismes, variërend van logische programma's en inductieve definities tot auto-epistemische en non-monotone logica's, zijn constructief van aard: ze beschrijven hoe we stap per stap bepaalde concepten kunnen construeren aan de hand van een procedure of een set regels.

Deze thesis bouwt een systematisch begrip op van constructieve kennis door te focussen op wat alle vormen van constructieve kennis samenbindt: het constructieproces. Dit werk richt zich naar 3 doelen: (1) het samenbrengen van verschillende vormen van constructieve kennis en het veralgemenen van de formalisatie van hun constructie processen, (2) het verduidelijken en uitbreiden van bestaande methoden om constructieve kennis te formaliseren en (3) het demonstreren van de de applicaties van deze abstracte formalisaties aan de hand van concrete voorbeelden uit KR.

We beginnen met het analyseren van constructieve definities en tonen aan dat, ondanks hun verschillende oorsprongen, de voorbeelden toch een gelijkaardig karakter vertonen: ze komen overeen met een *semantische operator* op een georderde *constructieruimte*. Het constructieproces wordt dan geformaliseerd als een monotone inductie van die operator. Verschillen tussen domeinen en soorten definities stammen voort uit de onderliggende constructie ruimte. Door deze verschillen expliciet te maken, kunnen we logica's ontwikkelen die diverse constructieve definities ondersteunen.

Voor non-monotone operatoren biedt Approximatie Fixpoint Theorie (AFT) een gefundeerd kader om de gewenste semantiek te vatten. Aan de ene kant formaliseert AFT het constructieproces van non-monotone operatoren aan de hand van goed gefundeerde inducties (well-founded inductions). Aan de andere kant biedt het een duidelijk algoritme om het juiste model te berekenen via vaste punt semantieken en monotone inducties van approximerende operatoren.

Ondanks dat AFT met veel success non-monotone inductie formaliseert, wordt de abstracte theorie maar beperkt opgenomen door de *Answer Set Programming (ASP)* en *Logic Programming (LP)* gemeenschappen. Deze thesis dicht dit gat door AFT te

herformuleren in termen bekend bij deze gemeenschappen. Daarmee demonstreren we de toepasbaarheid van AFT op het concrete probleem betreffende het uitbreiden van ASP tot programma's met aggregaten.

Verder breiden we ook AFT zelf uit door de noties van approximatie ruimtes en approximaties te versoepelen. Onze veralgemeende theorie (gAFT) brengt meerdere instantiaties met zich mee, allen gebaseerd op dezelfde fundamentele principes. De theorie behoudt de resultaten rond de vaste punt semantieken en de goed gefundeerde inducties uit de vorige versies en versterkt AFT door een grotere variatie van toepassingen en ruimtes te ondersteunen. We gebruiken concrete instantiaties, zoals AFT gebaseerd op *flowers* (fAFT) of intervallen (versoepelde iAFT), om de toepasbaarheid van de theorie in de verf te zetten aan de hand van concrete en praktische toepassingen zoals non-monotone recursieve definities, auto-epistemische logica's en LP.

Kortom, dit werk veralgemeent en verbindt eerder gefragmenteerde perspectieven op constructieve kennis. Het verduidelijkt bestaande methodes en breidt hen verder uit en het ondersteunt en demonstreert de abstracte theorie aan de hand van praktische en duidelijke voorbeelden uit het KR-domein.

List of Abbreviations

AFT Approximation Fixpoint Theory.

HT Logic of Here-and-There.

KK Kripke-Kleene (fixpoint).

ST Stable (fixpoint).

SUP Supported (fixpoint).

TSR Ternary Satisfaction Relation.

WF Well-Founded (fipxoint).

ALB Approximation Lower Bound.

AUB Approximation Upper Bound.

dAFT non-deterministic Approximation Fixpoint Theory.

fAFT flower Approximation Fixpoint Theory.

gAFT generalized Approximation Fixpoint Theory.

hAFT Higher order Approximation Fixpoint Theory.

iAFT interval Approximation Fixpoint Theory.

sAFT standard Approximation Fixpoint Theory.

ADF Abstract Dialectical Framework.

AEL Auto-Epistemic Logic.

AGI Artificial General Intelligence.

AI Artificial Intelligence.

ASP Answer Set Programming.

cpo chain-complete partial order.

DL Default Logic.

FO First Order Logic.

FO(ID) First Order Logic extended with Inductive Definitions.

KR Knowledge Representation.

LLM Large Language Model.

LP (Normal) Logic Programming.

SC Set Constraints (Programming).

wADF weighted Abstract Dialectical Framework.

List of Symbols

$(A \circ \iota)$ The natural extension of A

$(\mathcal{I}, \mathcal{J})$ A three- (or four-) valued interpretation represented by a pair of two-valued interpretations

$[a, b]_{\boldsymbol{\mathcal{L}}}$ The set of *ALBs* l between a and b, i.e., $a \preceq l \preceq b$

$[a, b]_{\boldsymbol{\mathcal{U}}}$ The set of *AUBs* u between a and b, i.e., $a \preceq u \preceq b$

$[h \mid t]$ A list with head h and tail t

$\boldsymbol{\mathcal{A}} \sim \boldsymbol{\mathcal{E}}$ $\boldsymbol{\mathcal{A}}$ approximates $\boldsymbol{\mathcal{E}}$; $\boldsymbol{\mathcal{A}}$ is an approximation space of $\boldsymbol{\mathcal{E}}$

$\boldsymbol{\mathcal{A}}$ An approximation space

$\boldsymbol{\mathcal{A}}_{triv}$ The trivial approximation space

$\boldsymbol{\mathcal{A}}_{\boldsymbol{f}}(\boldsymbol{\mathcal{P}})$ The approximation space in fAFT for the exact space $\boldsymbol{\mathcal{P}}$

$\boldsymbol{\mathcal{A}}_{\boldsymbol{i}}(\boldsymbol{\mathcal{P}})$ The approximation space in iAFT for the exact space $\boldsymbol{\mathcal{P}}$

$\mathscr{A}$ An approximation framework

$\langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ An approximation framework

Agg An aggregate symbol

α, β, γ Ordinals

A An approximating operator (approximator)

A_P An approximator of T_P

$A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}$ The first component of the approximator A_P derived from a lower-regular *TSR* $\models_3^{\boldsymbol{\mathcal{L}}}$

$A_P^{\models_3^{\mathcal{U}}}$ The second component of the approximator A_P derived from an upper-regular *TSR* $\models_3^{\mathcal{U}}$

$A^{\mathcal{L}}$ The projection of A on the *ALB*

$A^{\mathcal{L}}_{u^{\mathcal{U}}}$ The projection of $A([\![\boldsymbol{X}^{\mathcal{L}}, u]\!])$ on its *ALB*

$A^1_{x_2}(x_1)$ The projection of $A(x_1, x_2)$ on its first component in the context of sAFT or iAFT

$A^{\mathcal{U}}$ The projection of A on the *AUB*

$A^{\mathcal{U}}_{l^{\mathcal{L}}}$ The projection of $A([\![l, \boldsymbol{X}^{\mathcal{U}}]\!])$ on its *AUB*

$A^2_{x_1}(x_2)$ The projection of $A(x_1, x_2)$ on its second component in the context of sAFT or iAFT

$\boldsymbol{\mathcal{E}}(x^{\mathcal{A}})$ The set of exact approximants that approximate x

$Appr(\boldsymbol{\mathcal{A}})$ The set of approximators on $\boldsymbol{\mathcal{A}}$

$Appr(\boldsymbol{\mathcal{A}})$ The set of approximators over $\boldsymbol{\mathcal{A}}$

$Appr_O(\boldsymbol{\mathcal{A}})$ The set of approximators of O on $\boldsymbol{\mathcal{A}}$

$[\![.,.]\!]$ The recomposition function

$R^a_{\boldsymbol{X}}$ The set of application refinements of an approximant $\boldsymbol{X}$

$x^{\mathcal{A}}$ The set of approximants that approximate x

$\bot_{\mathcal{P}}$ The least element in $\mathcal{P}$

$\boldsymbol{\mathcal{F}}$ The cpo of flowers (in the context of a bounded-complete cpo)

$\circ$ The function composition operator

$+\!\!+$ The concatenation operator on lists

c_S A conjunctional atom for the set S

$\boldsymbol{\mathcal{C}}$ A construction space

ψ An embedding

$\boldsymbol{\mathcal{E}}$ An exact space

$\boldsymbol{\mathcal{E}}^2$ The product bilattice of the exact space $\boldsymbol{\mathcal{E}}$

$\mathcal{E}$ The ground set of the exact space $\boldsymbol{\mathcal{E}}$

$\mathcal{E}^2$ The set of pairs over the ground set $\mathcal{E}$

$\mathcal{E}^c$ The subset of consistent pairs of $\mathcal{E}^2$

$\mathcal{E}^e$ The subset of exact pairs of $\mathcal{E}^2$

$\boldsymbol{X}^{\boldsymbol{\mathcal{E}}}$ The set of exact elements approximated by $\boldsymbol{X}$

$\mathbf{f}$ The truth value 'false'

$\mathcal{F}$ The set of flowers (in the context of a bounded-complete cpo)

$\iota \circ A$ The natural abstraction of A

ι A monotonic function that entails a Galois connection between approximation spaces of an exact space

Γ_P A three-valued immediate consequence operator for the conjunctional program $P^\wedge$

Γ_P^C A three-valued immediate consequence operator for a condensed conjunctional program $P^\wedge(C)$

$\mathrm{gfp}(O)$ The greatest fixpoint of O

$\mathrm{glb}_{\boldsymbol{\mathcal{P}}}(S)$ The greatest lower bound of S in $\boldsymbol{\mathcal{P}}$

$R^g_{\boldsymbol{X}}$ The set of grounding refinements of an approximant $\boldsymbol{X}$

$\mathcal{G}$ A graph

$\mathcal{H}^2_{\mathcal{I}}(\phi)$ The (two-valued) truth value of a formula ϕ in an interpretation $\mathcal{I}$

$\mathcal{H}^3_{\mathfrak{A}}(\varphi)$ The (three-valued) truth value of a formula φ in a three-valued interpretation $\mathfrak{A}$

$\mathcal{H}^2$ A two-valued truth-function

$\mathcal{H}^3$ A three-valued truth function

$\mathcal{H}^{SK}$ Kleene's strong three-valued truth assignment

$\mathfrak{A}, \mathfrak{A}'$ Three- (or four-) valued (Herbrand) interpretation

Φ_P^{Aggr} A three-valued approximator associated with an aggrate-logic program P

Φ_P A three-valued immediate consequence operator of a program P

$\mathcal{I}, \mathcal{J}$ A two-valued (Herbrand) interpretation

$\mathcal{I}^\wedge, \mathcal{J}^\wedge$ Herbrand interpretations over a conjunctional extension $\Sigma^\wedge$

O	A(n) (semantic) operator
O^*	O lifted to approximants
$KK(A)$	The Kripke-Kleene fixpoint of A
λ	A limit ordinal or the lambda operator from lambda calculus
$\mathrm{lb}_{\boldsymbol{\mathcal{P}}}(S)$	The set of lower bounds of S in $\boldsymbol{\mathcal{P}}$
$S^{\downarrow}$	The downwards closure of a set S
$x^{\downarrow}$	The principal ideal of x; The set of all elements that are smaller than or equal to x
$\leq_{\mathbb{N}}$	The natural order on the natural numbers
$\preceq$	An order on the bounds $\boldsymbol{\mathcal{L}} \cup \boldsymbol{\mathcal{U}}$; the composition order
$\leq_a$	The acceptance order, in the context of wADFs
$\preceq_{\boldsymbol{\mathcal{E}}}$	A global approximant order
$\leq_d$	The (point-wise extension of) the definedness order
$\preceq_f$	The flower composition order
$\leq_i$	The information order
$\leq_p$	The precision order
$\leq_{pt}$	The pointwise extension of the acceptance order
$\leq_t$	The truth order
$\mathrm{lfp}(O)$	The least fixpoint of O
$\mathscr{L}$	A first-order logic
$.^{\boldsymbol{\mathcal{L}}}$	The lower decomposition function
$\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$	The *ALB* of an approximant $\boldsymbol{X}$
$S^{\boldsymbol{\mathcal{L}}}$	$.^{\boldsymbol{\mathcal{L}}}$ lifted to a set S of approximants
$\boldsymbol{\mathcal{L}}$	A cpo of *ALBs*; a lower decomposition space
$\mathcal{L}$	A set of *ALBs*
$\mathcal{L}_f$	The lower decomposition space in fAFT

$List$ The set of (finite and infinite) lists of natural numbers

$\mathrm{lub}_{\boldsymbol{\mathcal{P}}}(S)$ The least upper bound of S in $\boldsymbol{\mathcal{P}}$

$\mathcal{D}$ The defined concept

$\mathcal{X}, \mathcal{Y}$ Belief states

$Max_{\boldsymbol{\mathcal{P}}}(S)$ The set of maximal elements of $S \subseteq \boldsymbol{\mathcal{P}}$

$Min_{\boldsymbol{\mathcal{P}}}(S)$ The set of minimal elements of $S \subseteq \boldsymbol{\mathcal{P}}$

$\models_2$ An satisfaction relation for two-valued interpretations

$\models_3$ A ternary satisfaction relation

$\models_{FPL}$ The lower-regular *TSR* corresponding to the semantics provided by Faber et al. [41]

$\models_{F}$ The lower-regular *TSR* corresponding to the semantics provided by Ferraris [43]

$\models_{GL}$ The lower-regular *TSR* such that answer sets of programs induced by $\models_{GL}$ coincide with *GL*-answer sets

$\models_{GZ}$ The lower-regular *TSR* corresponding to the semantics provided by Gelfond and Zhang [47]

$\models_{LPST}$ The lower-regular *TSR* corresponding to the semantics provided by Liu et al. [57]

$\models_{MR}$ The lower-regular *TSR* corresponding to the semantics provided by Marek and Remmel [58]

$\models_{triv}$ The lower-regular *TSR* corresponding to the least precise regular truth assignment of Pelov et al. [69]

$\models_{ult}$ The lower-regular *TSR* corresponding to the most precise regular truth assignment of Pelov et al. [69]

$\models_{SK}$ The lower-regular *TSR* associated with the strong Kleene truth assignment

$\models_3^{\boldsymbol{\mathcal{L}}}$ A lower-regular *TSR*

$\models_3^{\boldsymbol{\mathcal{U}}}$ An upper-regular *TSR*

$\boldsymbol{\mathcal{O}}$ An object space

Φ A set of acceptance functions φ_s

$\boldsymbol{\mathcal{P}}$ A partially ordered set (poset)

2^S The power set lattice of a set S

π A projection function

$\mathcal{P}$ The ground set

$\boldsymbol{\mathcal{L}_A^X}$ The set of $\boldsymbol{X}$-complying *ALBs* with respect to A

$\boldsymbol{\mathcal{A}_s}$ An approximations space in sAFT

Σ An alphabet; a vocabulary; a set of propositional atoms

$\Sigma^\wedge$ The conjunctional extension of Σ

$\sim$ An 'approximates' relation

$ST(A)$ The set stable fixpoints of A

St^1 The first component of the stable revision operator

St The stable revision operator

St^2 The second component of the stable revision operator

$SUP(A)$ The set supported fixpoints of A

θ An injection

$\top_{\boldsymbol{\mathcal{P}}}$ The greatest element in $\boldsymbol{\mathcal{P}}$

$\top_{x\boldsymbol{\mathcal{A}}}$ The most precise approximant that approximates x

T_P The immediate consequence operator of a program P

t The truth value 'true'

$\langle \mathcal{A}, \leq_p \rangle$ An approximation space

$\langle \boldsymbol{\mathcal{E}}, \leq \rangle$ An exact space

$\langle \mathcal{E}^2, \leq_p, \leq_t \rangle$ The product bilattice of the exact space $\boldsymbol{\mathcal{E}}$

$\langle \mathcal{E}^2, \leq_p \rangle$ The approximation space of $\mathcal{E}$ in sAFT

$\langle \mathcal{L}, \preceq \rangle$ A cpo of *ALBs* $\mathcal{L}$ ordered by the composition order $\preceq$; a lower decomposition space

$\langle \mathcal{P}, \leq \rangle$ A partially ordered set (poset) with ground set $\mathcal{P}$ and partial order $\leq$

$\langle \mathcal{U}, \preceq \rangle$ A co-chain-complete cpo of *AUBs* $\mathcal{U}$ ordered by the composition order $\preceq$; an upper decomposition space

$\langle S, \Phi \rangle$ A wADF

$\mathrm{ub}_{\boldsymbol{\mathcal{P}}}(S)$ The set of upper bounds of S in $\boldsymbol{\mathcal{P}}$

$S^{\uparrow}$ The upwards closure of a set S

$x^{\uparrow}$ The principal filter of x; The set of all elements that are greater than or equal to x

U_O An ultimate approximator of O

u The truth value 'undefined'

$.^{\boldsymbol{\mathcal{U}}}$ The upper decomposition function

$\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$ The *AUB* of an approximant $\boldsymbol{X}$

$S^{\boldsymbol{\mathcal{U}}}$ $.^{\boldsymbol{\mathcal{U}}}$ lifted to a set S of approximants

$\boldsymbol{\mathcal{U}}$ A co-chain-complete cpo of *AUBs*; an upper decomposition space

$\mathcal{U}$ A set of *AUBs*

$\mathcal{U}_f$ The upper decomposition space in fAFT

φ, ψ A formula over a first-order logic

$WF(A)$ The well-founded fixpoint of A

Ξ A wADF

$\boldsymbol{X} \sim x$ $\boldsymbol{X}$ approximates x

$\boldsymbol{X}$ An approximant

$\boldsymbol{X}_{\varphi}$ The approximant that approximates all belief states where φ is satisfied

$\{\varphi\}^{\Sigma}$ The belief state consisting of all Herbrand interpretations over Σ that satisfy φ

a^{Aggr} An aggregate atom

B_S The combined rule bodies for S within the context of a logic program P

$body(r)$ The body of the rule r

D The defined object

$head(r)$ The head of the rule r

K	A modal operator indicating knowledge of the agent of an AEL theory
P	A program
$P^{\mathcal{J}}$	A reduct of P for an interpretation $\mathcal{J}$
$P^{\wedge}$	A conjunctional program
$P^{\wedge}(C)$	A condensed conjunctional logic program for a set C of conjunctional atoms
$R_{\boldsymbol{X}}$	The set of refinements of an approximant $\boldsymbol{X}$
S	A set
Y^X	The set of funtions from X to Y, i.e., with domain X and codomain Y

Contents

Abstract vii

Beknopte samenvatting ix

List of Abbreviations xii

List of Symbols xx

Contents xxi

List of Figures xxv

List of Tables xxvii

1 Introduction 1
- 1.1 A Need for Constructive Knowledge Representation . . . 1
- 1.2 Constructive Knowledge in KR . . . 3
- 1.3 The Construction Process . . . 4
- 1.4 Research Goals . . . 7
- 1.5 Structure of the Thesis . . . 9

2 Approximation Fixpoint Theory 11
- 2.1 Order Theory . . . 11
  - 2.1.1 Noteworthy Poset-elements . . . 13
  - 2.1.2 Noteworthy Subsets . . . 15
  - 2.1.3 Relating Posets . . . 19
  - 2.1.4 Types of Poset . . . 19
- 2.2 Fixpoint Theory . . . 23
- 2.3 Approximation Fixpoint Theory . . . 27
  - 2.3.1 Standard AFT . . . 30
  - 2.3.2 Interval AFT . . . 40

2.3.3 Other AFT Versions 41

3 Towards a Unifying View on Monotonic Constructive Definitions 43
3.1 Introduction 43
3.2 Algebraic Formalization 46
3.3 Different Flavors of Constructive Definitions 49
3.3.1 (Co-)inductive Definitions of Sets 49
3.3.2 (Co-)recursive Definitions of Functions 56
3.3.3 Definitions with Custom-Designed Cpo's 65
3.4 More Advanced Constructive Definitions 66
3.4.1 Simultaneous Construction 67
3.4.2 Nested Construction 68
3.5 Conclusion and Future Work 71

4 Analyzing Semantics of Aggregate Answer Set Programming 73
4.1 Introduction 73
4.2 Logic Programming and Approximation Fixpoint Theory 75
4.3 Ternary Satisfaction Relations 77
4.4 Generalizing the concept of answer set 81
4.5 Aggregate Programs in the AFT Framework 84
4.5.1 Approaches that fit in the AFT framework 86
4.5.2 Other Ternary Satisfaction Relations 89
4.5.3 Summary 94
4.6 Conclusions and future work 95

5 Consistent Approximation Fixpoint Theory with Refined Approximation Spaces 97
5.1 Introduction 97
5.1.1 Extending Consistent AFT 98
5.1.2 Generalizing Consistent AFT 102
5.1.3 Our Contributions 103
5.2 Preliminaries: AEL and wADFs 104
5.2.1 Auto-Epistemic Logic 104
5.2.2 Weighted Abstract Dialectical Frameworks 105
5.3 Approximations with Multiple Maximal Elements 106
5.4 Generalization of Consistent Approximation Spaces 111
5.5 Generalization of the Stable Revision Operator 125
5.5.1 Well-defined Stable Revision Operator 126
5.5.2 Monotonic Stable Revision Operator 132
5.5.3 Internal Stable Revision Operator on a Cpo 135
5.6 Generalization of Consistent AFT 138
5.6.1 Approximating an Exact Space 143
5.6.2 Relating Different Types of Fixpoints 153

5.6.3 Noteworthy Properties of Stable Fixpoints 155
5.7 Well-founded Inductions 157
5.7.1 A-Reliable and A-Prudent Refinements 161
5.7.2 Well-Founded Induction versus Well-Founded Fixpoint 162
5.7.3 Stable Fixpoint as a Terminal Approximant 163
5.8 Hierarchy of Approximation Spaces and Approximation Operators . . 164
5.8.1 Hierarchy in Expressiveness of Approximators 169
5.8.2 Hierarchy in Expressiveness of Approximation Spaces 171
5.9 Conclusion 175

6 Practical Applications of Generalized AFT and Flower-based AFT 179
6.1 Introduction 179
6.2 fAFT and Expressiveness in Auto-Epistemic Logic 181
6.3 Improved Flexibility with Respect to the Exact Space 183
6.3.1 Flower-AFT on Bounded Complete Cpos 184
6.3.2 Interval-based AFT on Bounded Complete Cpos 187
6.3.3 AFT on Monotonic Operators 189
6.3.4 AFT on Non-Monotonic Operators 191
6.4 fAFT and Normal Logic Programming 200
6.4.1 Ultimate Conjunctional Immediate Consequence Operator . . 202
6.4.2 Condensed Conjunctional Immediate Consequence Operators 211
6.5 Approximating across Non-Monotonic Reasoning Formalisms 218
6.6 Conclusion 222

7 Conclusion 225
7.1 Our Contributions 225
7.2 Future Directions 227

A gAFT and Necessary Conditions 229
A.1 Decomposition Spaces 230
A.2 Interlattice Properties 233

Bibliography 239

Curriculum Vitae 249

List of Figures

3.1 Monotonic inductions of the transitive closure of a graph 48
3.2 A monotonic induction for the set of terminating nodes of a graph . . 51
3.3 An induction process for the company shares problem. 56
3.4 The signature of the Fibonacci definition 58

4.1 The different *TSRs* ordered by $\leq_p$ 95

List of Tables

4.1 A summary of the different *TSRs* 94

Chapter 1

Introduction

1.1 A Need for Constructive Knowledge Representation

Knowledge Representation (KR) has long been a central area within Artificial Intelligence (AI). While much of today's AI landscape is dominated by data-driven approaches such as machine learning and large language models (LLMs), KR addresses a complementary challenge: how to formally capture, structure, and reason with information in a way that is transparent, explainable, and unambiguous. In the current era, this role is more relevant than ever. With the significant strides made by domains like generative AI, e.g., through the introduction of systems such as ChatGPT (Achiam et al. [2]), we are making real advancements towards achieving Artificial General Intelligence (AGI), i.e., a form of AI that matches and may even surpass human intelligence. However, while such models demonstrate remarkable fluency and adaptability, on their own, they have no notion of truth; they are prone to hallucinations [52] and may fail basic tasks. This is caused by the lack of underlying tools to explicitly reason about knowledge. This is where KR becomes crucial: it provides the formal tools to capture knowledge in a transparent and interpretable way. Moreover, it remains irreplaceable in domains where precise semantics and verifiable reasoning are crucial, such as law, medicine, or safety-critical systems.

The most elementary form of knowledge is simple factual statements, for instance '8 is a natural number' or 'Linde lives in Belgium'. These atomic facts directly express basic properties of objects. More complex forms of knowledge arise when relationships between concepts are articulated. For example, the statement 'every natural number is either even or odd' connects the notions of natural numbers, evenness, and oddness,

thereby expressing a general constraint on how these concepts interact. The information provided in our propositions serve as the resource for our reasoning processes, allowing us to derive new conclusions, detect inconsistencies, and extend our understanding of the domain under consideration.

A crucial observation is that knowledge often has a constructive character: it specifies how certain objects can be constructed. Consider, for example, the set of even natural numbers. One cannot memorize this infinite set in its entirety, but we can memorize a recipe that systematically enumerates its members:

- 0 is even, and
- if a number x is even, then so is $x + 2$.

These two rules suffice to construct the entire set. They also allow us to determine which objects do not belong to the set: 3 is not even, nor are π, Belgium, or the Dalai Lama.

Constructive knowledge is both ubiquitous and fundamental, and humans are remarkably adept at interpreting it. Most people can immediately understand the simple recipe described above and intuitively map it to the set of even numbers. Constructively, it has been argued that our basic cognitive skills for temporal and causal reasoning are key factors [16, 26, 28, 31]: our understanding of constructive knowledge stems from viewing the construction process as a causal process. We intuitively reason that each step 'causes' the next: 0 being even causes 2 to be even, 2 being even causes 4 to be even, and so on. This causal perspective helps us identify the intended set, distinguishing the even numbers from larger sets, such as all natural numbers, that also satisfy the rules but include elements that are not produced through the prescribed construction. In essence, humans naturally interpret constructive definitions as describing not just any set that satisfies the rules, but the set generated by the causal unfolding of the construction process. Alternatively, the defined set can be explained non-constructively as the least set that satisfies the rules.

Similar construction processes are found in a plethora of applications. For example, if we have a data-base of people and their parents, we can easily determine someone's ancestors. Specifically, your ancestors consist of your parents and all of their ancestors. Consequently, the ancestor-relation is constructed in terms of a parameter-concept, i.e., the parent-relation. Different data-bases for the parent-relation will produce different ancestor-relations without requiring changes to the definition.

In other words, constructive knowledge enables defining new concepts in terms of existing ones, allowing knowledge to be built incrementally and systematically. This approach introduces flexibility and reusability, since definitions can be applied across different contexts or parameterized by varying inputs. Moreover, it enables the

representation of concepts that are too large, complex, or abstract to describe directly, making it possible to reason about entities that cannot be enumerated explicitly.

Because constructive knowledge is so foundational, and because humans often reason about it intuitively, sometimes even unconsciously, it becomes crucial to formalize it. By providing formal representations, we can capture these intuitive constructions in precise terms, enabling us to reason about them systematically, verify properties of the constructions, and apply them in artificial intelligence and knowledge-based systems. In short, formalizing constructive knowledge transforms an inherently intuitive and causal understanding into a rigorous framework, making it possible to analyze, manipulate, and apply these constructions reliably across mathematics, logic, and AI.

1.2 Constructive Knowledge in KR

Constructive knowledge is not confined to a single discipline; rather, it is deeply embedded in mathematics, computer science, logic, and knowledge representation. Below, we highlight several areas where constructive knowledge plays a central role.

Inductive and recursive definitions are perhaps the most classical form of constructive knowledge, defining sets, relations, or functions in terms of simpler cases and rules for generating new ones. Inductive definitions of even natural numbers, the transitive closure of a graph, the satisfaction relation of first order logic (FO) illustrate this process: they describe how to build possibly infinite structures from finite rules. They have been the subject of longstanding interest, as evidenced by the work of Spector [82], Feferman [42], Martin-Löf [61], Moschovakis [63], Aczel [4]. In knowledge representation, logic programs can be understood as inductive definitions [33], where each rule specifies how new facts are derived from existing ones. This perspective is formalized by Denecker [26]'s extension of classical logic with inductive definitions FO(ID), which integrates both monotonic and non-monotonic induction. The intimate link between recursion, induction, and logic programming semantics is further elaborated by Paulson and Smith [66] in their studies of recursion in computational logic. It is the intuition of some researchers that inductive and recursive definitions form the declarative understanding of two well-known declarative programming paradigms, logic and functional programming [33, 51]. Moreover, due to the close connection between inductive definitions and causal information, inductive definitions are useful for expressing common sense causal knowledge [26, 28].

Recursive definitions are also crucial components of functional programming languages as supported by the work of Roberts [73], Soare [80], Rubio-Sanchez [75]. Functional programming languages such as Haskell and ML are built on inductively defined data types (e.g., natural numbers, lists, binary trees). Programs defined on these types are often recursive, and reasoning about their correctness depends on proofs by induction.

Proof by induction is essentially a constructive principle: by showing that a property holds for the base constructors and is preserved under recursive construction, one demonstrates that the property holds universally. Caldwell [20], for example, analyzes the principles of proofs by induction as they arise from functional definitions and shows how they are indispensable in program verification.

While induction defines sets and properties of finite objects (such as natural numbers, terms or arrays), another form of constructive definitions is better suited for defining properties of infinite objects (such as streams or infinite trees): co-inductive definitions. Co-recursive definitions allow the construction of entities such as streams, or automata. Co-induction provides the logical dual of induction, supporting reasoning through bi-simulation. Sangiorgi's influential work [77] formalizes this duality and demonstrates its importance in process algebras and non-well-founded set theory. Thus, co-induction generalizes the constructive paradigm to settings with 'infinite construction'.

Until now, we have mainly focused on constructive definitions of set-theoretical entities such as sets, relations, functions, lists and trees. However, within the KR field, we notice more intricate forms of constructive knowledge, e.g., epistemic belief states. Theories in auto-epistemic logic (AEL) and default logic (DL) are theories about the knowledge of an introspective agent who is able to reason about what it does or does not know and what is true. For example, if a friend had an accident and we do not know whether he was injured, we will be worried. Similarly, if we see a bird and we do not know that it cannot fly, we will assume that it can. The semantics of such a theory is determined by the state of the agent itself and this state is defined inductively as argued by Denecker et al. [35]. Similarly, abstract dialectical frameworks (ADFs) and their weighted extensions (wADFs) can be understood in terms of constructive processes for acceptance conditions of arguments, as shown in work by Brewka et al. [15] and [10].

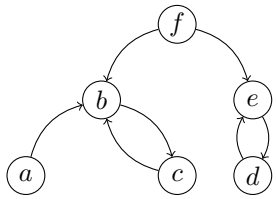
Underlying all these domains, whether in defining mathematical sets, reasoning about programs, proving properties of infinite structures, or modeling reflective forms of knowledge, the reasoning that takes place is about constructive knowledge, knowledge of how certain entities are to be constructed. Its ubiquity and diversity underline the need for general frameworks, such as approximation fixpoint theory, that can unify and extend constructive knowledge across different logics and applications.

1.3 The Construction Process

Having motivated the importance of constructive knowledge, we now turn our attention to the construction process itself. A closer examination of this process reveals how knowledge unfolds step by step. As a running illustrative example, let us consider the set of reachable nodes (from a specific node a) in a graph. We can constructively define this set R_a by the following rules

- a is reachable from a, and
- if there is an edge from a reachable node b to node c, then c is reachable as well.

Consider the following simple graph:



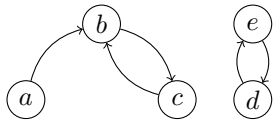
Intuitively, it is clear what the set of reachable nodes from a is: $R_a = \{a, b, c\}$. Indeed, this is what the construction process derives: we first use the first rule to derive that a is in our set, then we derive b and finally, we include c. Afterwards, we notice we cannot add any additional elements to our set, i.e., we have reached a fixpoint. Thus, the construction process corresponds to the following sequence:

$$\{\} \rightarrow \{a\} \rightarrow \{a, b\} \rightarrow \{a, b, c\}.$$

Interestingly, during the construction process, the information available at any intermediate stage reflects a lower bound of the set. What we have added to the set will remain in the set, but it is left completely open what we could still add to the set. This illustrates the inherently asymmetrical nature of such constructions, i.e., we accumulate certainty in one direction (what is definitely in the set), while leaving uncertainty in the other (what is definitely not in the set). In other words, when reasoning about these processes, we almost exclusively focus on the lower bound and tend to ignore potential upper bounds, i.e., the maximal sets of elements that could still conceivably be included given the current stage. Only at the end of the construction process, i.e., when a fixpoint is reached, does the upper bound collapse into the lower bound.

Nevertheless, in many cases it is just as straightforward to refine the upper bound as it is to refine the lower bound. Consider again the graph above: the node f has no incoming edges. It follows immediately that f is not reachable from any other node, and thus none of the construction rules will ever derive f. While this is evident from the outset, the construction process described earlier excludes f from the set only at the very final stage. Consequently, a formal system that restricts itself to tracking only lower bounds is forced to fully complete the construction before concluding that f is absent, even though this fact could have been established much earlier. This phenomenon is not limited to graph reachability. In general, constructive processes that disregard upper bounds risk inefficiency.

However, the asymmetry between lower and upper bounds in such construction process is not inherent, but rather a limitation of the way we formalize or think of them. Nothing is stopping us from also keeping track of the upper bounds during construction. Let us again look at the simple graph from before, but this time we will focus on the upper bounds. As mentioned before, we can safely remove f from the possible members of the set as it has no incoming edges. As a result, we are essentially left with the following graph:



Graphically, we can immediately observe that e and d will ultimately not belong to the reachable set, since each only has incoming edges from the other. However, at the current stage of the construction process, we still consider e and d as potentially belonging to the set. As a result, the second rule could still derive d based on e, and vice versa. Therefore, unlike the case of f, we cannot exclude d and e based on the lack of applicable rules. Instead, we recognize that d and e can be safely excluded from the set because, once both are removed, neither can ever be derived again by the rules. In summary, we can safely refine the upper bound as long as the remaining set is closed under rule application; that is, no rule can ever produce an element outside the set when applied to elements within it. This now leads to the following construction process of the upper bounds:

$$\{a, b, c, d, e, f\} \to \{a, b, c, d, e\} \to \{a, b, c\}$$

obtaining exactly the same set as identified by the construction process of the lower bounds. Thus, separately from each other, the two construction processes seemingly find the correct model.

The refining of upper bounds is strongly related to the notion of a ‘proof by induction’. In particular, when a property of an inductively defined set is proven by induction, we essentially impose an upper bound, corresponding to the property in question, and show that it is closed under rule application, i.e., it is an upper bound of everything.

This idea of including upper bounds as well as lower bounds during construction, is formalized by Approximation Fixpoint Theory (AFT), an abstract algebraic theory that has been very successful in providing semantics to different types of constructive knowledge. Here the constructive behavior is modeled by an approximator, i.e., an operator on the space of approximants (pairs of lower and upper bounds). In the terminology of AFT, the first type of refinement, which is driven by the direct application of rules, is referred to as an application refinement, while the second type,

which relies on ensuring that the set remains closed under all possible rule applications, is called a grounding refinement.

Interestingly, the upper bounds become essential when looking at non-monotonic rules, i.e., rules that may be applicable at one stage of the construction process, but whose applicability may be retracted in a later stage. A typical example of such definition is that of the satisfaction relation in first order logic (FO). The satisfaction relation specifies which logical formulas are true in which structures. For example, if in a structure $\mathcal{I}$, a and b are true, then $a \wedge b$ (a and b) is true and so is $a \vee b$ (a or b), in contrast $\neg a$ (not a) is not true in this structure. Besides a set of monotonic rules, the definition of the satisfaction relation has one non-monotonic rule:

- a formula $\neg\varphi$ is satisfied in a structure $\mathcal{I}$ if φ is not satisfied in $\mathcal{I}$.

Let $\mathcal{I}$ be the structure from before, where a and b hold. In the beginning of our construction process of this relation, we again start from the empty set as a lower bound. In this set, nothing is satisfied. If we were to apply the rule for negation, this would mean that for any possible formula φ, $\neg\varphi$ is satisfied in $\mathcal{I}$. This is clearly not what was intended. E.g., $\neg a$ or $\neg b$ are not actually satisfied by $\mathcal{I}$. Instead, we should only apply said rule when it has become clear that φ can never be derived. This is exactly what the upper bounds are able to capture. Thus, the upper bounds now capture information that is essential to construct the desired object.

Our knowledge is riddled with examples of non-monotonic definitions. For example, the shortest distance between two nodes in a graph can be modeled by a non-monotonic recursive definition. Similarly, access-control-policies where a person can grant or block access to a file, result in non-monotonic behaviors, and so on.

To conclude, construction spaces provide the natural setting in which constructive knowledge can be formally analyzed. Within this setting, AFT emerges as a particularly powerful tool: it equips us with a systematic framework to reason about both lower and upper bounds, and with this about monotonic and non-monotonic constructions. For these reasons, construction spaces and AFT will form key ingredients throughout the following chapters, serving as a foundation for our deeper exploration of constructive knowledge and its applications.

1.4 Research Goals

The overarching aim of this dissertation is to develop a deeper theoretical understanding of how we can formalize constructive knowledge, with a particular focus on the role of approximation spaces and construction processes. More concretely, the research has the following goals:

Analyzing and Generalizing Construction Processes. This thesis aims to investigate and formalize these types of constructive knowledge by means of rigorous mathematical frameworks. The central idea is to abstract away from the contingent details of specific examples or application domains, and instead to focus on what fundamentally unites them: their underlying construction processes. By isolating and studying the principles that govern such processes, we can identify the general mechanisms by which knowledge is built, extended, and constrained. This approach provides a unifying theoretical foundation for diverse instances of constructive knowledge, ranging from inductive and recursive definitions in mathematics to rule-based formalisms in knowledge representation and logic programming. Ultimately, the thesis seeks to show that by shifting our perspective from particular representations to the general theory of construction, we can better understand the underlying principles of constructive knowledge which is essential if we want to develop logics capable of handling them.

Clarifying and Extending Existing Tools. This work clarifies and elaborates on existing tools and proposed solutions, most prominently the framework of Approximation Fixpoint Theory, which has been widely recognized as a unifying semantic framework for reasoning about non-monotonic operators and constructive definitions. By revisiting AFT in the context of broader constructive knowledge, we highlight both its strengths and its limitations, and we demonstrate how they can be adapted or refined to better capture the nuances of different construction processes. In particular, on the one hand, this thesis aims at a better integration of AFT into the fields and domains where it can be applied. On the other hand, this work expands on the theory by exploring its limits. This improves our understanding of the underlying principles while also increasing expressiveness and usability of the theory, opening up a plethora of new avenues for future research.

Demonstrating Applications to Knowledge Representation. Another key objective of this thesis is to demonstrate how refined theoretical tools can be effectively applied in practice. To this end, we investigate their role in central knowledge representation settings such as recursive and co-recursive definitions, logic programming, and related KR formalisms. These case studies illustrate how the abstract principles of AFT not only provide a rigorous mathematical account of constructive processes, but also yield practical insights into the semantics of non-monotonic reasoning and complex dependencies.

By pursuing these objectives, the dissertation seeks to advance the study of constructive knowledge both conceptually and technically, and to show how a principled treatment of approximation spaces and construction spaces can enhance our ability to represent,

reason about, and compute with constructive forms of knowledge. This work has practical value in logic, artificial intelligence, and beyond.

1.5 Structure of the Thesis

In the next chapter we give all the necessary background information on order theory, fixpoint theory and approximation fixpoint theory, which are essential fields with respect to the following chapters.

The subsequent chapter investigates different types of constructive definitions arising from various domains and proposes a unifying formalization that reveals them as instances of the same underlying principles. Such a formalization is, on the one hand, a crucial step toward knowledge representation languages and logics that can accommodate a wide range of constructive definitions. On the other hand, by identifying both the commonalities and the differences among these definitions, we can better assess which research results transfer naturally across domains and where domain-specific challenges must be addressed.

The fourth chapter then turns to one specific type of construction, that present in logic programs, and specifically, in answer set programming (ASP). It shows how, using AFT, we can systematically define extensions of logic programming (LP) by focusing on the underlying constructive nature. For this, we look at different semantics for ASP with aggregates from literature. We analyze which of them follow the same fundamentally constructive principles, and thus fall into our framework, and which do not.

Then Chapters 5 and 6 explore the limits of AFT. Motivated by examples from various sources, these chapters aim to extend and improve AFT by allowing more flexible and refined approximants instead of simple pairs of lower bounds and upper bounds. Chapter 5 is a theoretical deep dive into the underlying principles of AFT. It identifies requirements we believe to be essential to the AFT-framework. Within this context it proposes a generalization of consistent approximation spaces and shows that this generalization is a conservative extension of AFT in every respect.

Chapter 6 focuses more on the practical implications of this theoretical extension. It ties back to the motivating examples and shows how the generalization not only improves on existing AFT with respect to expressiveness and usability, but also opens up new possibilities for an even more impressive potential to unify different semantics across different knowledge representation domains. At the same time it also identifies remaining gaps in the research.

The final chapter then concludes by summarizing the contributions and formulating potential future directions in this line of research.

Chapter 2

Approximation Fixpoint Theory

This chapter serves as a formal introduction to Approximation Fixpoint Theory (AFT). AFT, introduced by Denecker et al. [32], is a comprehensive constructive framework covering various semantics for non-monotonic reasoning formalisms in knowledge representation. In such reasoning formalisms, it is common practice and often very natural to associate a *semantic operator* on a space (*the exact space*) with a theory (or program). In case this operator is monotonic, the semantics will typically coincide with the least fixpoint of the operator. However, in general, the semantic operator is not monotonic. This severs the link between the intended semantics of the theory and the semantic operator. In fact, for many non-monotonic reasoning formalisms, different (disagreeing) semantics exist. This is where AFT comes in. Instead of the semantic operator, it uses an *approximating operator* and reduces different proposed semantics to different types of fixpoints of this approximating operator.

2.1 Order Theory

Order theory is a branch of mathematics that studies the ordering of elements within a set. In particular, it studies partially ordered sets (posets), i.e., sets of elements endowed with an order. This will prove crucial for analyzing the behavior of operators on these sets, which in turn enables the formalization of fixpoint theorems. This section delves deeper into the specifics of order theory, examining some key constructs and properties.

This foundation will pave the way for a thorough understanding of fixpoint theorems discussed in the next section.

Definition 2.1.1 (Partially Ordered Set (Poset))**.** *A* partially ordered set (poset) $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$ *is a set* $\mathcal{P}$ *(the* ground set*) equipped with a partial order* $\leq$*, i.e., a binary relation that satisfies*

- *reflexivity: for every* $x \in \mathcal{P}$*,* $x \leq x$*.*
- *anti-symmetry: for every* $x, y \in \mathcal{P}$*, if* $x \leq y$ *and* $y \leq x$*, then* $x = y$*.*
- *transitivity: for every* $x, y, z \in \mathcal{P}$*, if* $x \leq y$ *and* $y \leq z$*, then* $x \leq z$*.*

An example is the power set of a set S (the set of subsets of S), denoted by 2^S, with the partial order relation being the subset inclusion $\subseteq$. The pair $\langle 2^S, \subseteq \rangle$ is a poset.

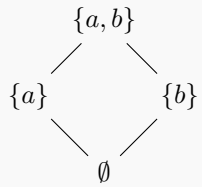
Definition 2.1.2 ((Induced) Subposet)**.** *A poset* $\boldsymbol{\mathcal{P}}^* = \langle \mathcal{P}^*, \leq^* \rangle$ *is a* subposet *of a poset* $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$ *if* $\mathcal{P}^* \subseteq \mathcal{P}$ *and* $\leq^* \subseteq \leq$*, i.e., if for some* $x, y \in \mathcal{P}^*$*,* $x \leq^* y$*, then* $x \leq y$*. A subposet is* induced *by* $\mathcal{P}^*$ *if for every* $x, y \in \mathcal{P}^*$*,* $x \leq^* y$ *iff* $x \leq y$*.*

To keep the notation simple and clean, often the same symbol will be used for the order on an induced subposet as for the original poset. Other examples of posets are those of the integers $\mathbb{Z}$ and natural numbers $\mathbb{N}$ endowed with the standard order, i.e., $\mathbb{Z} = \langle \mathbb{Z}, \leq_{\mathbb{Z}} \rangle$ and $\mathbb{N} = \langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$. Clearly $\mathbb{N}$ is a subposet of $\mathbb{Z}$ induced by $\mathbb{N}$. Both posets have the property that every two elements are comparable, a property that is known as *totality*.

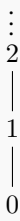
Definition 2.1.3 (Total Order)**.** *Let* $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$ *be a poset. The order* $\leq$ *is* total on a subset $S \subseteq \boldsymbol{\mathcal{P}}$ *if any pair of elements of* S *is comparable, i.e., if for every* $x, y \in S$*, either* $x \leq y$ *or* $y \leq x$*. The order* $\leq$ *is* total *if it is total on* $\boldsymbol{\mathcal{P}}$*.*

A poset is often visualized using a Hasse diagram. In such diagram, greater elements are placed above smaller elements to help visualizing the hierarchy. Moreover, edges are drawn between elements and their immediate predecessors and successors.

Example 2.1.4. *Take* $S = \{a, b\}$*, then the power set poset* $\langle 2^S, \subseteq \rangle$ *is visualized by the following Hasse diagram:*



Example 2.1.5. *Since the natural numbers poset* $\mathbb{N} = \langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$ *is infinitely large, only the smallest elements are visualized here:*



In general, the order in a poset gives rise to some interesting elements, operations and notions. For the remainder of this text we will use the symbol $\boldsymbol{\mathcal{P}}$ to denote a poset $\langle \mathcal{P}, \leq \rangle$.

2.1.1 Noteworthy Poset-elements

Definition 2.1.6 (Least and Greatest Element)**.** *An element* $x \in \boldsymbol{\mathcal{P}}$ *is*

- *a* least element of $\boldsymbol{\mathcal{P}}$ *if for every element* $y \in \boldsymbol{\mathcal{P}}$, $x \leq y$.
- *a* greatest element of $\boldsymbol{\mathcal{P}}$ *if for every element* $y \in \boldsymbol{\mathcal{P}}$, $y \leq x$.

If a least element, respectively a greatest element, exists, then it is unique and it is denoted by $\bot_{\boldsymbol{\mathcal{P}}}$*, respectively* $\top_{\boldsymbol{\mathcal{P}}}$*, where the subscript is dropped whenever the poset is clear from the context.*

Notice that not every poset has a least and greatest element, e.g., $\langle \mathbb{Z}, \leq_{\mathbb{Z}} \rangle$ has neither, while $\langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$ has a least element (0), but no greatest element. In contrast, the power set poset $\langle 2^S, \subseteq \rangle$ has both a least element ($\emptyset$) and a greatest element (S).

Definition 2.1.7 (Lower and Upper Bounds)**.** *Let* $S \subseteq \boldsymbol{\mathcal{P}}$*, then* $x \in \boldsymbol{\mathcal{P}}$ *is*

- *an* upper bound *of S if for every $s \in S$, $s \leq x$.*
- *a* lower bound *of S if for every $s \in S$, $x \leq s$.*

We use $\mathrm{lb}_{\boldsymbol{\mathcal{P}}}(S)$ and $\mathrm{ub}_{\boldsymbol{\mathcal{P}}}(S)$ to respectively denote the set of lower bounds and upper bounds of S in $\boldsymbol{\mathcal{P}}$.

Definition 2.1.8 (Greatest Lower and Least Upper Bound)**.** *Let $S \subseteq \boldsymbol{\mathcal{P}}$, then $x \in \boldsymbol{\mathcal{P}}$ is*

- *the* least upper bound *of S, denoted by $\mathrm{lub}_{\boldsymbol{\mathcal{P}}}(S)$, if it is the least of all upper bounds of S, i.e., $x \in \mathrm{ub}_{\boldsymbol{\mathcal{P}}}(S)$ and $x \leq u$ for each $u \in \mathrm{ub}_{\boldsymbol{\mathcal{P}}}(S)$.*
- *the* greatest lower bound *of S, denoted by $\mathrm{glb}_{\boldsymbol{\mathcal{P}}}(S)$, if it is the greatest of all lower bounds of S, i.e., $x \in \mathrm{lb}_{\boldsymbol{\mathcal{P}}}(S)$ and $l \leq x$ for every $l \in \mathrm{lb}_{\boldsymbol{\mathcal{P}}}(S)$.*

To simplify notation, we will drop the subscript $\boldsymbol{\mathcal{P}}$ whenever the poset is clear from the context.

Alternatively, the least upper bound of a set S is referred to as the *join* or the *supremum* of S and is denoted by $\bigvee S$ (or $x \vee y$ in case $S = \{x, y\}$). Similarly, the greatest lower bound of S is also called the *meet* or the *infimum* of S and is often denoted by $\bigwedge S$ (or $x \wedge y$ if $S = \{x, y\}$). Note that the least upper bound or greatest lower bound of S might not exist.

Definition 2.1.9 (Minimal and Maximal Elements)**.** *Let $S \subseteq \boldsymbol{\mathcal{P}}$, then*

- *the* set of minimal elements of *S, is defined as*

$$Min(S) := \{x \in S \mid \neg\exists y \in S : y < x\}.$$

- *the* set of maximal elements of *S, is defined as*

$$Max(S) := \{x \in S \mid \neg\exists y \in S : x < y\}.$$

.

Definition 2.1.10 (Principal Ideal and Principal Filter of an Element)**.** *The* principal ideal of an element *$x \in \boldsymbol{\mathcal{P}}$ is defined as $x^{\downarrow} := \{y \in \boldsymbol{\mathcal{P}} \mid y \leq x\}$. Similarly, the* principal filter of an element *$x \in \boldsymbol{\mathcal{P}}$ is given by $x^{\uparrow} := \{y \in \boldsymbol{\mathcal{P}} \mid x \leq y\}$.*

Informally, the principal ideal of an element $x \in \boldsymbol{\mathcal{P}}$ is the set of elements of $\boldsymbol{\mathcal{P}}$ that are smaller than x. In other words, $x^{\downarrow}$ is the greatest subset of $\boldsymbol{\mathcal{P}}$ for which x is a (least) upper bound. Analogously, $x^{\uparrow}$ is the greatest subset of $\boldsymbol{\mathcal{P}}$ such that x is a (greatest) lower bound. An analogous concept can be defined for sets.

Definition 2.1.11 (Downwards and Upwards Closure of a Set)**.** *The* downwards closure of a set $S \subseteq \mathcal{P}$ *is defined as*

$$S^{\downarrow} := \{y \mid y \leq x \text{ for some } x \in S\}.$$

Similarly, the upwards closure of a set $S \subseteq \mathcal{P}$ *is given by*

$$S^{\uparrow} := \{y \mid x \leq y \text{ for some } x \in S\}.$$

From this definition, it trivially follows that $S^{\downarrow} = \bigcup\{x^{\downarrow} \mid x \in S\}$ and $S^{\uparrow} = \bigcup\{x^{\uparrow} \mid x \in S\}$. Hence, $x^{\downarrow} = \{x\}^{\downarrow}$ and $x^{\uparrow} = \{x\}^{\uparrow}$.

2.1.2 Noteworthy Subsets

Definition 2.1.12 (Chain)**.** *A set $S \subseteq \mathcal{P}$ is* a chain *if $\leq$ is total on S.*

For instance, since $\leq_{\mathbb{N}}$ is total, *any* subset S of $\mathbb{N}$ is a chain in $\langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$. On the other hand, $\{\emptyset, \{a\}, \{a, b\}\}$ and $\{\emptyset, \{a, b\}\}$ are examples of chains in the power set poset $\langle 2^{\{a,b\}}, \subseteq \rangle$ while $\{\emptyset, \{a\}, \{b\}\}$ is not a chain.

Definition 2.1.13 (Saturated Chain)**.** *A chain $C \subseteq \mathcal{P}$ is* saturated *if it is a maximal totally ordered subset of $\mathcal{P}$, i.e., if for every chain $C' \subseteq \mathcal{P}$, if $C \subseteq C'$, then $C = C'$.*

For example, since $\leq_{\mathbb{N}}$ is total, $\langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$ has only one saturated chain, i.e., $\mathbb{N}$. In comparison, $\langle 2^{\{a,b\}}, \subseteq \rangle$ has two saturated chains, in particular $\{\emptyset, \{a\}, \{a, b\}\}$ and $\{\emptyset, \{b\}, \{a, b\}\}$.

Definition 2.1.14 (Anti-Chain)**.** *A set $S \subseteq \mathcal{P}$ is* an anti-chain *if every two distinct elements of S are incomparable, i.e., if for every $x, y \in S$, if $x \neq y$, then $x \not\leq y$ and $y \not\leq x$.*

Clearly, for any poset $\mathcal{P}$, every set that contains strictly less than two distinct elements is an anti-chain. This consists of the empty set $\emptyset$ and all singletons $\{x\}$ with $x \in \mathcal{P}$. In case $\leq$ is a total order, these are the only anti-chains of $\mathcal{P}$. If $\leq$ is not a total order, then it must induce at least one other anti-chain. For instance, $\langle 2^{\{a,b\}}, \subseteq \rangle$ gives rise to the additional anti-chain $\{\{a\}, \{b\}\}$.

Later in this chapter, the notion of chains is used to distinguish between different types of posets. In particular, we are interested in the existence and inclusion of the least upper bound and greatest lower bound of the chains. Sometimes we refer to these bounds as *the limits of the chains* hinting at the strong connection between chains and sequences as described below. First, we consider a useful subclass of chains that straightforwardly map to sequences.

Definition 2.1.15 (Well-ordered set)**.** *A set $S \subseteq \mathcal{P}$ is* well-ordered *if each non-empty subset $S' \subseteq S$ has a least element.*

Note that a well-ordered set S is a chain in $\mathcal{P}$ and that each non-greatest element $x \in S$ must have a successor, i.e., the least element of $(x^{\uparrow} \cap S) \backslash \{x\}$. For example, $\mathbb{N} \subseteq \langle \mathbb{Z}, \leq_{\mathbb{Z}} \rangle$ is a well-ordered set, while $\mathbb{Z}$ is not.

It is important to notice that any well-ordered set S can be ordered into a strictly increasing (or strictly decreasing) sequence $(x_i)_{i<\lambda}$ with λ an ordinal. Naturally, if such an increasing (respectively decreasing) sequence $(x_i)_{i<\lambda}$ has a limit in $\mathcal{P}$, then it must be the least upper bound $\mathrm{lub}(S)$ (respectively greatest lower bound $\mathrm{glb}(S)$). To derive a similar result for chains that are not well-ordered, we need cofinality.

Definition 2.1.16 (Cofinal Subset)**.** *A set $S' \subseteq S \subseteq \mathcal{P}$ is a* cofinal subset *of S if for every $s \in S$ there exists an $s' \in S'$ such that $s \leq s'$.*

By definition, if a set S has a greatest element $\top_S$, then the smallest cofinal set of S is $\{\top_S\}$. Linking back to the example of natural numbers and integers above, $\mathbb{N}$ is a cofinal subset of $\mathbb{Z} \subseteq \langle \mathbb{Z}, \leq_{\mathbb{Z}} \rangle$. Hence, the chain $\mathbb{Z}$ in $\langle \mathbb{Z}, \leq_{\mathbb{Z}} \rangle$ has a well-ordered cofinal subset. Interestingly, this property is generalized for arbitrary chains in the following proposition, which is a well known application of Zorn's lemma, and a consequence of the axiom of choice.

Proposition 2.1.17 (From [49], Theorem 1.5, page 72)**.** *If $S \subseteq \mathcal{P}$ is a chain, then S has a well-ordered cofinal subset.*

Proposition 2.1.18. *If S' is a cofinal subset of $S \subseteq \mathcal{P}$, then $x \in \mathcal{P}$ is an upper bound of S iff x is an upper bound of S'.*

Proof. Since $S' \subseteq S$, it trivially follows that if x is an upper bound of S, then x is an upper bound of S' by definition of an upper bound.

On the other hand, if x is an upper bound of S', then for every $s' \in S'$, $s' \leq x$. Moreover, since S' is a cofinal subset of S, for every $s \in S$ there exists an $s' \in S'$ such that $s \leq s'$, hence $s \leq x$. Therefore x is also an upper bound of S. □

In other words, any chain $S \subseteq \mathcal{P}$ has a well-ordered subset S' such that $\mathrm{lub}(S) = \mathrm{lub}(S')$. Hence, there exists an increasing sequence $(x_i)_{i<\lambda}$ with limit $\mathrm{lub}(S)$ such that for every $i < \lambda$, $x_i \in S$. Analogously, by inverting the order in the poset, there exists a decreasing sequence $(x_i)_{i<\lambda}$ with limit $\mathrm{glb}(S)$ such that for every $i < \lambda$, $x_i \in S$.

Definition 2.1.19 (Closed Under Limits of Chains (Closed set))**.** *The set $S \subseteq \mathcal{P}$ is* closed under limits of chains *if for every non-empty chain $C \subseteq S$,*

- *if* $\mathrm{glb}(C)$ *exists, then* $\mathrm{glb}(C) \in S$.
- *if* $\mathrm{lub}(C)$ *exists, then* $\mathrm{lub}(C) \in S$.

To simplify the terminology, we will often use the term closed set *to denote a set which is closed under limits of chains.*

Definition 2.1.20 (Convex Set)**.** *A set* $S \subseteq \boldsymbol{\mathcal{P}}$ *is* convex *if for every* $x, z \in S$ *and* $y \in \boldsymbol{\mathcal{P}}$ *such that* $x \leq y \leq z$*, it holds that* $y \in S$.

For example, the set $\{2, 3, 4\}$ in the poset $\langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$ is closed and convex. On the other hand, let us assume that $\langle \mathbb{R}, \leq_{\mathbb{R}} \rangle$ is the poset of the real numbers under the usual order. Then, the set $\{2, 3, 4\} \subseteq \langle \mathbb{R}, \leq_{\mathbb{R}} \rangle$ is closed, but not convex. In contrast, the open interval $(2, 4)$ is convex, but not closed since, e.g., $\mathrm{lub}(2, 4) = 4$ is not included in the interval. Furthermore, the set $(2, 4)\backslash\{3\}$ is neither closed, nor convex. With regards to this work, we are only interested in closed sets within posets where each non-empty chain has a least upper bound and greatest lower bound. Intuitively, one can think of a closed set S in such posets as a set that is bounded by its minimal and maximal elements, i.e., for every element $x \in S$ there must exist an $m \in \mathit{Min}(S)$ and an $n \in \mathit{Max}(S)$ such that $m \leq x \leq n$. On the other hand, a convex set S is a set that can be completely characterized by its lower and upper closure. Hence, a closed and convex set in such posets is completely characterized by its minimal and maximal elements. These intuitions are formalized in the following propositions.

Proposition 2.1.21. *If* $S \subseteq \boldsymbol{\mathcal{P}}$ *is closed and for every non-empty chain* $S' \subseteq S$, $\mathrm{lub}(S')$ *and* $\mathrm{glb}(S')$ *exist, then* $S^{\downarrow} = \mathit{Max}(S)^{\downarrow}$ *and* $S^{\uparrow} = \mathit{Min}(S)^{\uparrow}$.

Proof. We only prove the first equality here, the second is completely analogous.

It trivially follows from the definition of the lower closure that $\mathit{Max}(S)^{\downarrow} \subseteq S^{\downarrow}$ since $\mathit{Max}(S) \subseteq S$.

For the other direction, take an arbitrary element $x \in S^{\downarrow}$, we need to show that there exists an $m \in \mathit{Max}(S)$ such that $x \leq m$. By definition of $S^{\downarrow}$ there must exist a $y \in S$ such that $x \leq y$. Now take the set $y^{\uparrow} \cap S$. This set contains y. Take a saturated chain $R \subseteq y^{\uparrow} \cap S$. Since R is saturated, it must contain y. Thus, R is a non-empty chain in S, thus $m = \mathrm{lub}(R)$ exists. Moreover, $x \leq y \leq m$. Furthermore, S is a closed set, therefore $m \in S$.

Let us assume that $m \notin \mathit{Max}(S)$, then there must exist an $m' \in S$ such that $m < m'$. Since $y \in R$, $y \leq m < m'$, hence $m' \in y^{\uparrow} \cap S$. Moreover, for every $r \in R$, $r \leq m < m'$, thus $R \cup \{m'\}$ is a chain. This contradicts the earlier assumption that R is saturated, thus $m \in \mathit{Max}(S)$.

It then trivially follows that $x \in Max(S)^{\downarrow}$. Since x was chosen arbitrarily, this holds for every $x \in S^{\downarrow}$, thus $S^{\downarrow} \subseteq Max(S)^{\downarrow}$, hence $S^{\downarrow} = Max(S)^{\downarrow}$. □

Proposition 2.1.22. *If $S \subseteq \mathcal{P}$, then $S^{\downarrow}$ and $S^{\uparrow}$ are convex sets.*

Proof. Let $x, z \in S^{\downarrow}$ and $y \in \mathcal{P}$ such that $x \leq y \leq z$. Since $z \in S^{\downarrow}$, there must exist an $s \in S$ such that $y \leq z \leq s$. Therefore $y \in S^{\downarrow}$. Similarly, let $x, z \in S^{\uparrow}$. Since $x \in S^{\uparrow}$, there must exist an $s \in S$ such that $s \leq x \leq y$. Therefore, $y \in S^{\uparrow}$. Hence, $S^{\downarrow}$ and $S^{\uparrow}$ are convex. □

Proposition 2.1.23. *If $S \subseteq \mathcal{P}$ is convex, then $S = S^{\downarrow} \cap S^{\uparrow}$.*

Proof. Clearly, by definition of the downwards and upwards closure of a set, $S \subseteq S^{\downarrow}$ and $S \subseteq S^{\uparrow}$, thus $S \subseteq S^{\downarrow} \cap S^{\uparrow}$.

On the other hand, take an arbitrary element $x \in S^{\downarrow} \cap S^{\uparrow}$. Since $x \in S^{\uparrow}$, there must exist an $m \in S$ such that $m \leq x$. Similarly, since $x \in S^{\downarrow}$, there must exist an $n \in S$ such that $x \leq n$. Hence, there exist a pair $m, n \in S$ such that $m \leq x \leq n$. Since S is convex, it then follows that $x \in S$. Since x was chosen arbitrarily, this must hold for every $x \in S^{\downarrow} \cap S^{\uparrow}$, thus $S^{\downarrow} \cap S^{\uparrow} \subseteq S$. Therefore, $S = S^{\downarrow} \cap S^{\uparrow}$. □

Corollary 2.1.24. *If $S \subseteq \mathcal{P}$ is closed and convex, and for every chain $S' \subseteq S$, $\mathrm{glb}(S')$ and $\mathrm{lub}(S')$ exist, then $S = Max(S)^{\downarrow} \cap Min(S)^{\uparrow}$.*

Proposition 2.1.25. *If S is a set of convex sets over $\mathcal{P}$, then the intersection $\bigcap(S)$ is convex.*

Proof. Let $x, z \in \bigcap(S)$ and $y \in \mathcal{P}$ such that $x \leq y \leq z$. Since $x, z \in \bigcap(S)$, for every $X \in S$, $x, z \in X$. Since X is convex, also $y \in X$. This is the case for every $X \in S$, therefore $y \in \bigcap(S)$. Thus, $\bigcap(S)$ is convex. □

Proposition 2.1.26. *If S is a set of closed sets over $\mathcal{P}$, then the intersection $\bigcap(S)$ is closed.*

Proof. Let $C \subseteq \bigcap(S)$ be a non-empty chain. Then for every $X \in S$, C is a chain in X. Since X is closed, this implies that if $\mathrm{glb}(C)$ exists, then $\mathrm{glb}(C) \in X$, and analogously, if $\mathrm{lub}(C)$ exists, then $\mathrm{lub}(C) \in X$. This is the case for any $X \in S$, therefore if $\mathrm{glb}(C)$ exists, then $\mathrm{glb}(C) \in \bigcap(S)$, and if $\mathrm{lub}(C)$ exists, then $\mathrm{lub}(C) \in \bigcap(S)$. Consequently, $\bigcap(S)$ is closed. □

Definition 2.1.27 (Lower-Bounded or Upper-Bounded Set)**.** *A set $S \subseteq \mathcal{P}$ is* lower-bounded *if $\mathrm{glb}(S)$ exists and $\mathrm{glb}(S) \in S$. Similarly, S is* upper-bounded *if $\mathrm{lub}(S)$ exists and $\mathrm{lub}(S) \in S$.*

2.1.3 Relating Posets

Sometimes posets may show very similar structures, despite having completely different underlying ground sets. To compare posets across ground sets, we consider the well-known notions of embeddings and isomorphisms.

Definition 2.1.28 (Embedding)**.** *A function $\psi : \mathcal{P}_1 \hookrightarrow \mathcal{P}_2$ is an* embedding *from poset $\mathcal{P}_1$ to $\mathcal{P}_2$ if*

$$\forall x, y \in \mathcal{P}_1 : x \leq_1 y \Leftrightarrow \psi(x) \leq_2 \psi(y)$$

We say $\mathcal{P}_1$ is embedded *in $\mathcal{P}_2$.*

Definition 2.1.29 (Isomorphism)**.** *A function $\psi : \mathcal{P}_1 \to \mathcal{P}_2$ is an* isomorphism *between posets $\mathcal{P}_1$ and $\mathcal{P}_2$ if ψ is a bijective embedding. We say $\mathcal{P}_1$ is* isomorphic *to $\mathcal{P}_2$.*

2.1.4 Types of Poset

Very few restrictions are imposed on the notion of a poset. This results in a broad range of posets. In general, fixpoint theories require more restrictive classes of posets. These additional constraints give rise to interesting properties which enable the formalization of useful fixpoint theories. This work considers four important and interesting classes of posets, namely (1) *chain-complete posets* or *cpos*, (2) *cochain-complete posets*, (3) *bounded-complete cpos*, and (4) *complete lattices*.

Definition 2.1.30 (Chain-Complete Poset (cpo))**.** *A poset $\mathcal{P}$ is* a chain-complete partial order (cpo) *if every chain of $\mathcal{P}$ has a least upper bound.*

Note that every cpo has a least element $\bot$, which is the least upper bound of the empty chain $\emptyset$. The dual concept of a cpo is a cochain-complete poset.

Definition 2.1.31 (Cochain-Complete Poset)**.** *A poset $\mathcal{P}$ is* a cochain-complete partial order *if every chain of $\mathcal{P}$ has a greatest lower bound.*

Similarly to how cpos have a least element, cochain-complete partial orders have a greatest element $\top$, which is the greatest lower bound of the empty chain $\emptyset$. It is easy to see that if $\mathcal{P}$ is chain-complete, then the poset $\mathcal{P}'$ obtained from $\mathcal{P}$ by inverting the order is cochain-complete, and vice versa. We can restrict the class of cpos even further, for example by requiring bounded-completeness.

Definition 2.1.32 (Bounded-Complete cpo)**.** *A cpo $\mathcal{P}$ is* bounded-complete *if every non-empty subset $S \subseteq \mathcal{P}$ has a greatest lower bound* $\mathrm{glb}(S) \in \mathcal{P}$.

Proposition 2.1.33. *If $\mathcal{P}$ is a bounded-complete cpo and $S \subseteq \mathcal{P}$ has an upper bound, then S has a least upper bound.*

Proof. Take the set of upper bounds Q of S, then $Q \subseteq \mathcal{P}$ and Q is non-empty. Since $\mathcal{P}$ is bounded-complete, Q must have a greatest lower bound $\mathrm{glb}(Q) \in \mathcal{P}$.

Take an arbitrary $s \in S$. Since Q is a set of upper bounds of S, it must hold that $s \leq q$ for every $q \in Q$. Hence, s is a lower bound of Q, therefore $s \leq \mathrm{glb}(Q)$ by definition of the greatest lower bound. Since s was chosen arbitrarily, this holds for every $s \in S$, thus $\mathrm{glb}(Q)$ is an upper bound of S, i.e., $\mathrm{glb}(Q) \in Q$. In conclusion, $\mathrm{glb}(Q)$ is the least upper bound of S. □

Finally, the most restrictive of the four classes are the complete lattices. AFT as introduced by Denecker et al. [32] only covers operators on complete lattices.

Definition 2.1.34 (Complete Lattice)**.** *A* complete lattice *is a poset $\mathcal{P}$ such that every subset S of $\mathcal{P}$ has a least upper bound* $\mathrm{lub}(S)$ *and a greatest lower bound* $\mathrm{glb}(S)$.

A complete lattice has a least element $\bot$ and a greatest element $\top$, i.e., the least upper bound and greatest lower bound of the empty set. Clearly, by definition, each complete lattice is a (bounded-complete) cpo and a chain-complete poset.

Proposition 2.1.35. *If $\mathcal{P}$ is a bounded-complete cpo with a greatest element $\top$, then $\mathcal{P}$ is a complete lattice.*

Proof. Since $\mathcal{P}$ is bounded-complete, every non-empty subset S of $\mathcal{P}$ has a greatest lower bound. Moreover, since $\mathcal{P}$ has a greatest element, this is the greatest lower bound of the empty set.

Clearly for every $S \subseteq \mathcal{P}$, the greatest element $\top$ is an upper bound of S. Then by Proposition 2.1.33, S has a least upper bound. □

Corollary 2.1.36. *If $\mathcal{P}$ is a bounded-complete cpo and $\mathcal{P}$ is cochain-complete, then $\mathcal{P}$ is a complete lattice.*

Proposition 2.1.37 (From [24], Theorem 2.31, page 47)**.** *$\mathcal{P}$ is a complete lattice iff $\mathcal{P}$ has a least element and every non-empty subset $S \subseteq \mathcal{P}$ has a least upper bound in $\mathcal{P}$. Alternatively, $\mathcal{P}$ is a complete lattice iff $\mathcal{P}$ has a greatest element and every non-empty subset $S \subseteq \mathcal{P}$ has a greatest lower bound in $\mathcal{P}$.*

Example 2.1.38 (Example 2.1.4, contd.)**.** *The power set poset $\langle 2^{\{a,b\}}, \subseteq \rangle$ is a complete lattice. In fact, for any set S, the power set poset $\langle 2^S, \subseteq \rangle$ is a complete lattice. In particular, the greatest lower bound of any family F of subsets of S is given by the intersection of those sets, i.e., $\bigcap F$. Similarly, the least upper bound of F corresponds to the union $\bigcup F$.*

Example 2.1.39 (Example 2.1.5, contd.)**.** *In contrast, the poset* $\mathbb{N} = \langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$ *is not even a cpo or a cochain-complete cpo. For example, the chain* $\mathbb{N}$ *does not have a least upper bound. Similarly, the poset does not have a greatest element, thus the empty chain* $\emptyset$ *does not have a greatest lower bound.*

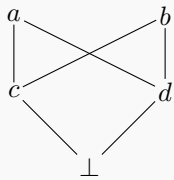
Oftentimes, when a poset does not satisfy the necessary requirements, it can be extended with *virtual elements* to fit in the desired class. In this case, we can extend the natural numbers poset with a greatest element infinity (∞).

Example 2.1.40. *Let* $\langle \mathbb{N}^\infty, \leq_{\mathbb{N}^\infty} \rangle$ *be the poset of the natural numbers extended with a top element* ∞*. This poset is a cpo, now* $\mathrm{lub}(\mathbb{N})$ *is* ∞*. Moreover,* $\langle \mathbb{N}^\infty, \leq_{\mathbb{N}^\infty} \rangle$ *is also cochain-complete. In fact, the extended cpo is a complete lattice.*

In particular, the greatest lower bound of any non-empty set is given by its unique minimal element. Similarly, the least upper bound of any non-empty set is given by its unique maximal element if it exists, and by ∞ *otherwise. On the other hand, the greatest lower bound of the empty set is* ∞*, while the least upper bound of the empty set is* 0*.*

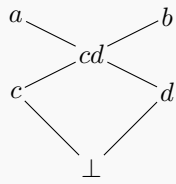
Let us now consider the following posets to further clarify the distinction and connections between the four classes.

Example 2.1.41. *It is easy to check that the following poset is a cpo, it has a least element and every non-empty chain has a least upper bound, e.g.,* $\mathrm{lub}(\{a, c\}) = a$*.*



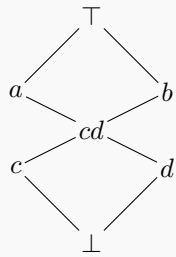
While the poset in Example 2.1.41 is a cpo, it is not bounded-complete. Specifically, the set $\{a, b\}$ has three lower bounds c, d and $\bot$, but no greatest lower bound. However, it is possible to extend the cpo with a single element such that the extended cpo is bounded-complete.

Example 2.1.42. *After adding the element cd, the following cpo is bounded-complete. Now the greatest lower bound of $\{a, b\}$ is cd.*



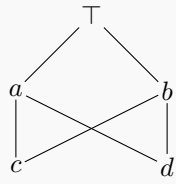
Although the cpo above is bounded-complete, it is still not a complete lattice. For instance the set $\{a, b\}$ does not have a least upper bound. This is easily solved by adding a greatest element $\top$ to the poset.

Example 2.1.43. *By adding a top-element $\top$, we obtain a complete lattice. Now the least upper bound of $\{a, b\}$ is $\top$.*



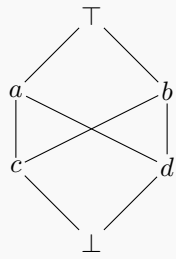
Until now, we only discussed examples of cochain-complete posets which are complete lattices. However, not every cochain-complete poset is a complete lattice. In fact, not every cochain-complete poset is a cpo.

Example 2.1.44. *It is easy to check that the following poset is cochain-complete, it has a greatest element and every non-empty chain has a greatest lower bound, e.g., $\mathrm{glb}(\{a, c\}) = c$.*



Clearly, this poset is not a cpo, since it does not have a least element.

Example 2.1.45. *By adding a least element $\bot$, we obtain a cochain-complete cpo.*



Although the previous poset is a cochain-complete and chain-complete, it is not a complete lattice. E.g., $\{a, b\}$ does not have a greatest lower bound, furthermore $\{c, d\}$ does not have a least upper bound. Hence, the class of complete lattices is a strict subset of the cochain-complete cpos.

2.2 Fixpoint Theory

Whereas posets provide useful information on the space on which an operator operates, fixpoint theory, on the other hand, identifies interesting elements in the space related to the operator itself. In particular, it concerns elements which are mapped to themselves by the operator, known as fixpoints.

Definition 2.2.1 ((Pre-/Post-)Fixpoint)**.** *Let $O : \mathcal{P} \to \mathcal{P}$ be an operator on $\mathcal{P}$. An element $x \in \mathcal{P}$ is*

- *a* fixpoint *of O if $O(x) = x$.*
- *a* pre-fixpoint *of O if $O(x) \leq x$.*
- *a* post-fixpoint *of O if $x \leq O(x)$.*

Research is often especially interested in the least and greatest fixpoint since these tend to correspond to the intended semantics associated with the operator.

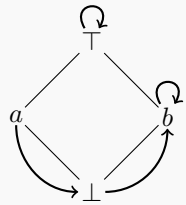
Definition 2.2.2 (Least and Greatest Fixpoints)**.** *Let $O : \mathcal{P} \to \mathcal{P}$ be an operator on $\mathcal{P}$ and let $x \in \mathcal{P}$ be a fixpoint of O, then*

- *x is the least fixpoint of O, denoted by $\mathrm{lfp}(O)$, if it is smaller than any other fixpoint of O, i.e., if for every fixpoint y of O, $x \leq y$.*

- *x is the greatest fixpoint of O, denoted by $\mathrm{gfp}(O)$, if it is greater than any other fixpoint of O, i.e., if for every fixpoint y of O, $y \leq x$.*

Least and greatest pre- and post-fixpoints are defined analogously.

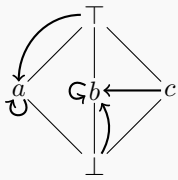
Example 2.2.3. *Consider the following operator O on a simple complete lattice. The operator is visualized by arrows, e.g., $O(a) = \bot$.*



The fixpoints of O are b and $\top$, specifically b is the least fixpoint, while $\top$ is the greatest fixpoint. On the other hand, a is a pre-fixpoint and $\bot$ is a post-fixpoint.

Note that the least and greatest fixpoint do not always exist. It is also not always the case that every element in the space is either a pre-fixpoint, post-fixpoint or both.

Example 2.2.4. *The following operator does not have a least fixpoint or greatest fixpoint. Instead, its fixpoints a and b are incomparable. Moreover, c is neither a pre-fixpoint, nor a post-fixpoint.*



Although the existence of a least fixpoint or greatest fixpoint is not guaranteed in general, when the operator is monotonic (i.e., the operator preserves the order), it can be assured for various classes of posets as formalized by a multitude of theorems of varying forms.

Definition 2.2.5 (Monotonic Operator)**.** *An operator $O \colon \mathcal{P} \to \mathcal{P}$ on a poset $\mathcal{P}$ is* monotonic *if for all $x, y \in \mathcal{P}$ such that $x \leq y$, it holds that $O(x) \leq O(y)$.*

Definition 2.2.6 (Anti-monotonic Operator)**.** *An operator $O \colon \mathcal{P} \to \mathcal{P}$ on a poset $\mathcal{P}$ is* anti- monotonic *if for all $x, y \in \mathcal{P}$ such that $x \leq y$, it holds that $O(x) \geq O(y)$.*

Now we are ready to discuss the foundational Knaster-Tarski theorem [84], which guarantees the existence of the least and greatest fixpoints for monotonic operators on complete lattices.

Theorem 2.2.7 (Knaster-Tarski Theorem [84])**.** *If $\mathcal{P}$ is a complete lattice and the operator $O\colon \mathcal{P} \to \mathcal{P}$ is monotonic, then the set of fixpoints of O, equipped with $\leq$, forms a complete lattice. In particular, the least and greatest fixpoint coincide with the least pre-fixpoint and the greatest post-fixpoint, respectively.*

Since the set of fixpoints corresponds to a complete lattice, there must exist a least element, i.e., the least fixpoint, and a greatest element, i.e., the greatest fixpoint. Later, Markowsky obtained a similar result for the more general class of cpos [60].

Theorem 2.2.8 (Markowsky [60])**.** *If $\mathcal{P}$ is a cpo and O is a monotonic operator on $\mathcal{P}$, then the set of fixpoints of O, equipped with $\leq$, forms a cpo. Moreover, the least fixpoint coincides with the least pre-fixpoint.*

A cpo does not necessarily have a greatest element, therefore, the existence of a greatest fixpoint is not guaranteed. Cousot and Cousot [23] provide a constructive version of the Knaster-Tarski theorem. In their work, Cousot and Cousot [23] show that the least fixpoint of a monotonic operator on a complete lattice can be constructed as the limit of a (possibly transfinite) sequence. Interestingly, their proof of this aspect does not strictly require the poset to be a complete lattice, instead it suffices to work in a cpo.

Theorem 2.2.9 (Cousot and Cousot [23])**.** *If $\mathcal{P}$ is a cpo and O is a monotonic operator on $\mathcal{P}$, then the least fixpoint can be constructed as the limit of a possibly transfinite sequence $(x_\alpha)_{\alpha<\beta}$ with β an ordinal, and*

- $x_{\alpha+1} = O(x_\alpha)$*, and*
- $x_\lambda = \mathrm{lub}(\{x_\alpha \mid \alpha < \lambda\})$ *for limit ordinals λ.*

This constructive approach to finding the least fixpoint of a monotonic operator was further generalized by Denecker and Vennekens [30] to allow for more flexibility in the sequence through the notion of *monotonic inductions*. This generalization mimics human reasoning capabilities more closely, e.g. in the context of constructive definitions. Chapter 3 elaborates on this further. The following theorem was originally only formulated for complete lattices, but it can easily be generalized to cpos instead.

Definition 2.2.10 ((Terminal) monotonic Induction)**.** *Let O be a monotonic operator on a cpo. A* monotonic induction *is an increasing sequence $(x_\alpha)_{\alpha<\beta}$ satisfying*

- $x_\alpha \leq x_{\alpha+1} \leq O(x_\alpha)$*, and*

- $x_\lambda = \text{lub}(\{x_\alpha \mid \alpha < \lambda\})$, *for limit ordinals* $\lambda \leq \beta$ *(in particular,* $x_0 = \bot$*).*

A monotonic induction is terminal *if its limit* x_β *cannot be refined further, i.e., if* $x_\beta \not< O(x_\beta)$.

We will refer to the monotonic induction identified by Cousot and Cousot [23] as the *fastest induction*.

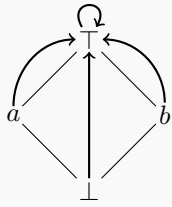
Theorem 2.2.11 (Denecker and Vennekens [30])**.** *All terminal monotonic inductions of a monotonic operator* O *on a cpo* $\mathcal{P}$ *converge (in the limit) to the least fixpoint of* O*, i.e., if* $(x_\alpha)_{\alpha<\beta}$ *is a terminal monotonic induction, then* $x_\beta = \text{lfp}(O)$.

Proof. First we show that for every $\alpha < \beta$, $x_\alpha \leq \text{lfp}(O)$. This follows from transfinite induction. Clearly $x_0 = \bot \leq \text{lfp}(O)$. Assume $x_\alpha \leq \text{lfp}(O)$, then $x_{\alpha+1} \leq O(x_\alpha) \leq O(\text{lfp}(O)) = \text{lfp}(O)$ where the last inequality follows from the monotonicity of O. Assume for a limit ordinal λ that for every $\alpha < \lambda$, $x_\alpha \leq \text{lfp}(O)$, then by definition of a least upper bound, $x_\lambda = \text{lub}(\{x_\alpha \mid \alpha < \lambda\}) \leq \text{lfp}(O)$. Consequently $x_\beta \leq \text{lfp}(O)$.

Now we only need to prove that x_β is a fixpoint of O, then also $\text{lfp}(O) \leq x_\beta$. For this we only need to prove that for every $\alpha < \beta$, $x_\alpha \leq O(x_\alpha)$, then by monotonicity of O we can conclude that $x_\beta \leq O(x_\beta)$. Since $(x_\alpha)_{\alpha<\beta}$ is terminal, this entails that $x_\beta = O(x_\beta)$, thus x_β is a fixpoint of O.

Clearly $x_0 = \bot \leq O(x_0)$. Assume that $x_\alpha \leq O(x_\alpha)$. We know that $x_\alpha \leq x_{\alpha+1} \leq O(x_\alpha)$. Then by monotonicity of O, $O(x_\alpha) \leq O(x_{\alpha+1})$, hence $x_{\alpha+1} \leq O(x_{\alpha+1})$. On the other hand, assume for a limit ordinal λ that for every $\alpha < \lambda$, $x_\alpha \leq O(x_\alpha)$, then $x_\lambda = \text{lub}(\{x_\alpha \mid \alpha < \lambda\}) \leq \text{lub}(\{O(x_\alpha) \mid \alpha < \lambda\})$. Moreover, for every $\alpha < \lambda$, $x_\alpha \leq x_\lambda$. Therefore, by monotonicity of O, $O(x_\alpha) \leq O(x_\lambda)$. Thus, $O(x_\lambda)$ is an upper bound of $\{O(x_\alpha) \mid \alpha < \lambda\}$. Hence, $x_\lambda \leq \text{lub}(\{O(x_\alpha) \mid \alpha < \lambda\}) \leq O(x_\lambda)$. □

Example 2.2.12. *The following is a monotonic operator on a simple complete lattice where everything is mapped to the greatest element:*



Clearly, $\top$ is the only fixpoint, thus also the least and greatest fixpoint. The least fixpoint $\top$ can be obtained constructively by, e.g., the following monotonic induction:

$$\bot \rightarrow \bot \rightarrow a \rightarrow \top$$

Note that the aforementioned monotonic induction is not the only possible induction, nor is it the most efficient induction of the operator. Instead, the fastest monotonic induction *of O is given by:*

$$\bot \rightarrow \top$$

Proposition 2.2.13. *Let $\mathcal{P}^*$ be an induced subposet of a cpo $\mathcal{P}$, and let O be a monotonic function on $\mathcal{P}$ which is internal on $\mathcal{P}^*$. If $\mathcal{P}^*$ is a cpo that coincides with $\mathcal{P}$ for the least upper bounds of its chains, then O has a least fixpoint in $\mathcal{P}$, specifically, the least fixpoint of O in $\mathcal{P}^*$.*

Proof. As O is a monotonic function on $\mathcal{P}$ which is internal on $\mathcal{P}^*$, the restriction of O to $\mathcal{P}^*$ is a monotonic operator. Moreover, $\mathcal{P}^*$ is a cpo, thus by Theorem 2.2.9, O has a least fixpoint in $\mathcal{P}^*$ which can be constructed using a terminal monotonic induction $(x_\alpha)_{\alpha<\beta}$. Let f be a fixpoint of O in $\mathcal{P}$. Now we will show that for every $\alpha < \beta$, $x_\alpha \leq f$. For limit ordinals this follows from the assumption that the posets coincides for the least upper bounds of their chains, for successor ordinals, this follows from the monotonicity of O. □

2.3 Approximation Fixpoint Theory

Here, we recall the basics of AFT from the work by Denecker et al. [32, 36]. Oftentimes, in knowledge representation, a theory defines a semantic operator O on a complete lattice $\boldsymbol{\mathcal{E}} = \langle \mathcal{E}, \leq \rangle$. If O is monotonic, usually the semantics of the theory corresponds to the least fixpoint of O. The previous section concluded that such a fixpoint must exist and that it can be obtained constructively. In contrast, when the semantic operator is non-monotonic, the existence of this fixpoint is not guaranteed. Moreover, when it exists, it does not always correspond to the intended semantics. In fact, for many non-monotonic reasoning logics, different (disagreeing) semantics exist. This is where AFT can be applied. AFT is an abstract lattice theoretic formalization of constructive methods for non-monotonic operators. It defines different types of constructions and fixpoints for an approximator in an approximation space, including supported (SUP), Kripke-Kleene (KK), stable (ST) and well-founded (WF) fixpoints. The theory has been applied to a range of non-monotonic logics to characterize existing as well as new semantics: e.g., LP and ASP [37], auto-epistemic and default logic [35], higher order LP [22, 12], argumentation frameworks and abstract dialectal frameworks [83, 10]. To

clarify many of the relatively abstract key concepts of AFT, this section will discuss examples within the following context of a simple access control system.

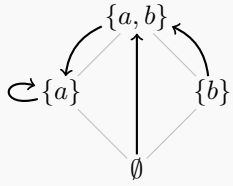
Example 2.3.1. *Consider the following simple access control system for shared files:*

- *A person has access to a file if they are granted access and they are not blocked from having access to said file.*
- *Other than the owner of a file, only people with access to a file can grant or block access to said file.*

Note that there are no rules prohibiting a person from giving access to themselves. Assume the system is used by a set of colleagues S. Given an access policy between those colleagues, their system administrator wants to determine who has access to a file. This type of problem gives rise to operators on the power set lattice $\boldsymbol{\mathcal{E}} = \langle 2^S, \subseteq \rangle$. A set $T \in \boldsymbol{\mathcal{E}}$ represents the situation where everyone in T has access to the file, while everyone else has not.

Clearly this access control system quickly gives rise to non-monotonic operators.

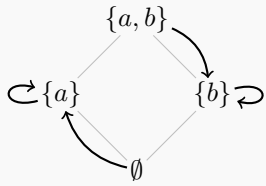
Example 2.3.1.a. *Assume S consists of two colleagues, i.e., Alice (a) and Bob (b). Consider the access policy where Alice always blocks Bob's access, while Bob always grants Alice access. Assume that the owner of the file grants both of them access to the file. This can be represented by the following operator O_1:*



It is easy to see that O_1 is non-monotonic. For example, while $\emptyset \subseteq \{a, b\}$, $O_1(\emptyset) = \{a, b\} \not\subseteq \{a\} = O_1(\{a, b\})$. Still, O_1 has a least fixpoint. Specifically, O_1 has only one fixpoint, i.e., $\{a\}$, hence $\{a\}$ is trivially the least fixpoint.

Irregardless of the previous example, unlike in the monotonic case, the least fixpoint is no longer guaranteed to exist, nor does it necessarily capture the intended semantics. Consequently, our interest shifts from the least fixpoint to well-founded or stable fixpoints, which avoid self-supportedness as illustrated by the following example.

Example 2.3.1.b. *Alternatively, consider the access policy where Bob always blocks Alice's access while granting himself access. Suppose the owner of the file granted Alice access. This can be represented by the following operator O_2:*



Clearly O_2 is again non-monotonic. In particular, whether Alice has access to the file depends on Bob not *having access to the file. For example, while $\{a\} \subseteq \{a, b\}$, $O_2(\{a\}) = \{a\} \not\subseteq \{b\} = O_2(\{a, b\})$. The operator has two fixpoints $\{a\}$ and $\{b\}$, but no least fixpoint. Note that both fixpoints are minimal.*

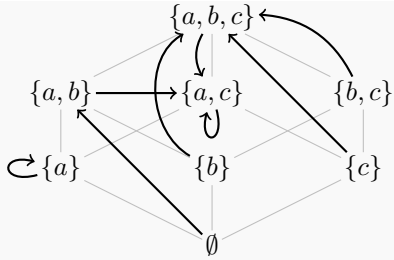
While $\{b\}$ is also a minimal fixpoint, it is clear that Bob will only have access to the file if he has access to the file. Thus, there are no grounds for Bob having access. We say the access of Bob is unfounded and the fixpoint $\{b\}$ is self-supported. In contrast, $\{a\}$ is well-founded and corresponds to the intended semantics where Alice is the only person with access to the file.

While both previous examples correspond to non-monotonic operators, they still allow us to derive the expected fixpoints by iterated application of the operators. The next example shows that this is not the case in general.

Example 2.3.1.c. *Assume that Alice and Bob have a third colleague Charles (c) and that the owner grants access to Alice and Bob. Consider the following access policy:*

- *Alice always blocks Bob's access.*
- *Bob always grants Charles access.*
- *Charles always grants himself access.*

This situation can be represented by the following operator O_3:



Again, this operator is non-monotonic. For example, while $\emptyset \subseteq \{a\}$, $O_3(\emptyset) = \{a, b\} \not\subseteq \{a\} = O_3(\{a\})$.

While O_3 *has a least fixpoint* $\{a\}$*, iteratively applying* O_3 *obtains the other non-minimal fixpoint* $\{a, c\}$*. However, similarly to Example 2.3.1.b, the access of Charles is unfounded. In this fixpoint, Charles only has access because he has access. Instead, the system administrator should arrive at the conclusion that only Alice should have access.*

This final example clearly showcases why we cannot rely on iterative application of a non-monotonic operator. Specifically, such an operator might jump too high and then get stuck in an undesirable fixpoint. AFT avoids this pitfall by only considering what is certain. More precisely, AFT starts from a state of complete uncertainty, and gradually refines our knowledge. This knowledge is encapsulated by approximations, and specifically, by lower and upper bounds.

2.3.1 Standard AFT

The first step of AFT is to associate an approximation space $\mathcal{A}$ with the relevant complete lattice $\mathcal{E}$. For this, Denecker et al. [32] consider the product bilattice of $\mathcal{E}$.

Definition 2.3.2 (Product Bilattice)**.** *The* product bilattice $\mathcal{E}^2$ *of a complete lattice* $\mathcal{E}$ *is the set of pairs over* $\mathcal{E}$*, denoted by* $\mathcal{E}^2$*, equipped with a precision order* $\leq_p$ *and truth order* $\leq_t$ *where*

- $(x_1, x_2) \leq_p (y_1, y_2)$ *iff* $x_1 \leq y_1$ *and* $y_2 \leq x_2$.
- $(x_1, x_2) \leq_t (y_1, y_2)$ *iff* $x_1 \leq y_1$ *and* $x_2 \leq y_2$.

Then, the term *approximation space* specifically refers to the poset $\mathcal{A}_s = \langle \mathcal{E}^2, \leq_p \rangle$. It is easy to show that this poset is a complete lattice.

Proposition 2.3.3. *If $\mathcal{E}$ is a complete lattice, then $\mathcal{A}_s = \langle \mathcal{E}^2, \leq_p \rangle$ is a complete lattice as well.*

Proof. Let $S \subseteq \mathcal{A}_s$. Then $S_1 = \{s_1 \mid (s_1, s_2) \in S\}$ is a subset of $\mathcal{E}$, therefore the least upper bound $\mathrm{lub}(S_1)$ and the greatest lower bound $\mathrm{glb}(S_1)$ exist. Similarly, for $S_2 = \{s_2 \mid (s_1, s_2) \in S\}$, $\mathrm{lub}(S_2)$ and $\mathrm{glb}(S_2)$ exist. Then it is clear that for every pair $(s_1, s_2) \in S$, $\mathrm{glb}(S_1) \leq s_1$ and $s_2 \leq \mathrm{lub}(S_2)$, thus $(\mathrm{glb}(S_1), \mathrm{lub}(S_2)) \leq_p (s_1, s_2)$, therefore $(\mathrm{glb}(S_1), \mathrm{lub}(S_2))$ is a lower bound of S. Analogously we can conclude that $(\mathrm{lub}(S_1), \mathrm{glb}(S_2))$ is an upper bound of S.

Assume an arbitrary lower bound $(t_1, t_2) \in \mathcal{A}_s$ of S, then we find that for every $(s_1, s_2) \in S$, $t_1 \leq s_1$ and $s_2 \leq t_2$. Since this holds for any $s_1 \in S_1$ and for any $s_2 \in S_2$, this entails that $t_1 \leq \mathrm{glb}(S_1)$ and $\mathrm{lub}(S_2) \leq t_2$, therefore $(t_1, t_2) \leq_p (\mathrm{glb}(S_1), \mathrm{lub}(S_2))$, hence $\mathrm{glb}(S) = (\mathrm{glb}(S_1), \mathrm{lub}(S_2))$. Analogously, we find that $\mathrm{lub}(S) = (\mathrm{lub}(S_1), \mathrm{glb}(S_2))$. □

Throughout this work, other variants of AFT, in particular with different approximation spaces, are discussed. Therefore, to distinguish the original version of AFT from the others, it is referred to as standard AFT (sAFT) in the remainder of the text.

Definition 2.3.4. *An approximation pair $(x_1, x_2) \in \mathcal{A}_s$* approximates *all elements of $\mathcal{E}$ for which x_1 is a lower bound and x_2 is an upper bound, i.e., a pair $(x_1, x_2) \in \mathcal{A}_s$ approximates any $x \in \{y \mid x_1 \leq y \leq x_2\} \subseteq \mathcal{E}$.*

Sometimes we will also refer to the approximation pair with the more general term *approximant* which abstracts away from the concrete approximation space that is used. Intuitively, we can think of such approximants as intervals. Specifically, a pair $(x_1, x_2) \in \mathcal{A}_s$ is an approximation of any $x \in [x_1, x_2]$. However, this intuition assumes that the pair is consistent.

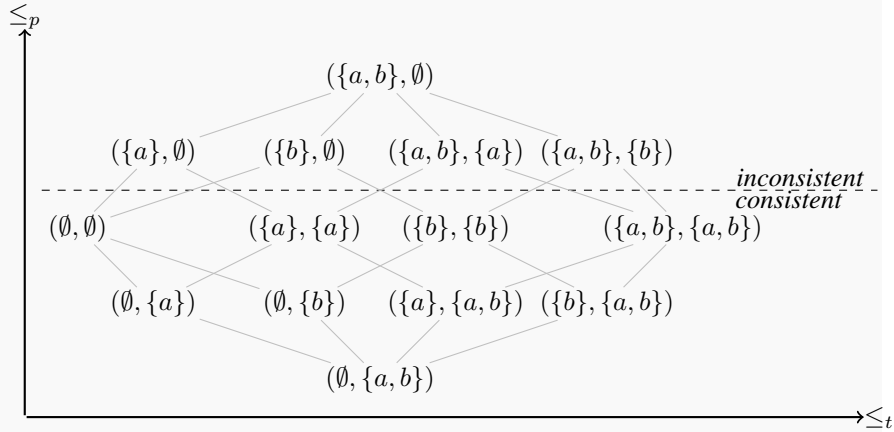
Definition 2.3.5 (Consistent Pair)**.** *A pair $(x_1, x_2) \in \mathcal{A}_s$ is* consistent *if $x_1 \leq x_2$. The set of consistent pairs is denoted by $\mathcal{E}^c$, i.e., $\mathcal{E}^c = \{(x_1, x_2) \in \mathcal{E}^2 \mid x_1 \leq x_2\}$.*

Clearly, if a pair is not consistent, then it does not approximate any element $x \in \mathcal{E}$ and vice versa. Another special set of pairs are those that approximate exactly one element of $\mathcal{E}$.

Definition 2.3.6 (Exact Pair)**.** *A pair $(x_1, x_2) \in \mathcal{A}_s$ is* exact *if it approximates exactly one element, i.e., if $x_1 = x_2$. The set of exact pairs is denoted by $\mathcal{E}^e$, i.e., $\mathcal{E}^e = \{(x_1, x_2) \in \mathcal{E}^2 \mid x_1 = x_2\}$.*

Under the truth order the exact pairs essentially form an embedding of $\mathcal{E}$ in $\mathcal{E}^2$. Accordingly, the space $\mathcal{E}$ is sometimes referred to as the *exact space*. It trivially follows that every exact pair is also consistent.

Example 2.3.7 (Example 2.3.1, contd.)**.** *For the examples where $S = \{a, b\}$, the bilattice[1] is visualized by the diagram below. The precision order is visualized vertically (as in standard Hasse diagrams), while the truth order is visualized horizontally, i.e., if some (x_1, x_2) is positioned to the right of some (y_1, y_2), then $(x_1, x_2) \leq_t (y_1, y_2)$.*



A pair (x_1, x_2) in the bilattice represents the knowledge that at least all members of x_1 and at most all members of x_2 have access to the file. This knowledge is only consistent if x_1 is a subset of x_2, i.e., $x_1 \subseteq x_2$. Everything below the dashed line is consistent while everything above the line is inconsistent. Immediately below the dashed line are the exact pairs which, under the truth order, form an embedding of the original poset $\boldsymbol{\mathcal{E}}$. In this case, an exact pair (x, x) corresponds to the knowledge that the set of people with access to the file is exactly x, e.g., the pair $(\{a\}, \{a\})$ exclusively approximates the situation in which only Alice has access.

The second step of AFT is to associate an approximating operator A on the approximation space, which approximates the behavior of O. Essentially, the approximating operator uses the available approximate knowledge, i.e., knowledge about the lower and upper bound, to derive a lower and upper bound for the results of O.

[1] We have dropped the pairs with incomparable lower and upper bounds, i.e., $(\{a\}, \{b\})$ and $(\{b\}, \{a\})$, from our visualization to avoid overcomplicating the diagram. These elements are not relevant as they are never considered during construction.

Definition 2.3.8 (Approximating Operator (Approximator))**.** *An operator $A : \boldsymbol{\mathcal{A}_s} \to \boldsymbol{\mathcal{A}_s}$ is an* approximating operator *or* approximator *if*

- *A is $\leq_p$-monotonic, i.e., $(x_1, x_2) \leq_p (y_1, y_2)$ implies $A(x_1, x_2) \leq_p A(y_1, y_2)$, and*
- *A is symmetric, i.e., $A(x_1, x_2) = (y_1, y_2)$ iff $A(x_2, x_1) = (y_2, y_1)$.*

The set of approximators on $\boldsymbol{\mathcal{A}_s}$ is denoted by $Appr(\boldsymbol{\mathcal{A}_s})$.

Importantly, the monotonicity means that increasing the precision of the input to an approximator A increases the precision of the output, or informally, more precise information will obtain more precise results. While the work by Denecker et al. [32] only considers symmetric approximators, the authors claim this requirement is technically not essential. However, as we will discuss later in this section, approximators showcase additional desirable properties when subject to the constraint of symmetry. Therefore, we choose to stay close to the original definition of approximators and also restrict it to symmetric approximators on $\boldsymbol{\mathcal{A}_s}$.

Definition 2.3.9. *An approximator $A : \boldsymbol{\mathcal{A}_s} \to \boldsymbol{\mathcal{A}_s}$* approximates *an operator $O : \boldsymbol{\mathcal{E}} \to \boldsymbol{\mathcal{E}}$ if for every $(x_1, x_2) \in \boldsymbol{\mathcal{A}_s}$ and $y \in \boldsymbol{\mathcal{E}}$, $A(x_1, x_2)$ approximates $O(y)$ if (x_1, x_2) approximates y.*

Intuitively, $A(x_1, x_2)$ approximates the behavior of O on the interval $[x_1, x_2]$.

Definition 2.3.10. *An approximator $A : \boldsymbol{\mathcal{A}_s} \to \boldsymbol{\mathcal{A}_s}$* extends *an operator $O : \boldsymbol{\mathcal{E}} \to \boldsymbol{\mathcal{E}}$ if for every $x \in \boldsymbol{\mathcal{E}}$, $A(x, x) = (O(x), O(x))$.*

Proposition 2.3.11. *An approximator $A : \boldsymbol{\mathcal{A}_s} \to \boldsymbol{\mathcal{A}_s}$ approximates an operator $O : \boldsymbol{\mathcal{E}} \to \boldsymbol{\mathcal{E}}$ iff A extends O.*

Proof. Assume that $A(x, x) = (y_1, y_2)$. If A approximates O, then we know that since x is approximated by (x, x), $O(x)$ is approximated by $A(x, x) = (y_1, y_2)$, thus $y_1 \leq O(x) \leq y_2$. Since A is symmetric, it also holds that $A(x, x) = (y_1, y_2)$, therefore $y_1 = y_2$, hence $y_1 = y_2 = O(x)$. Therefore A extends O.

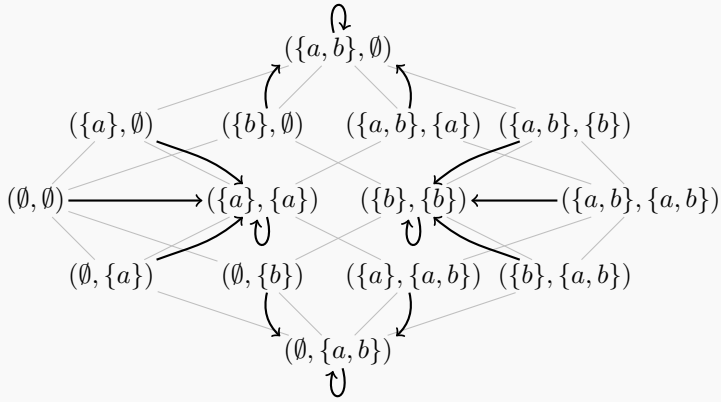
Assume an arbitrary approximating pair (x_1, x_2). For every y approximated by (x_1, x_2), it holds that $(x_1, x_2) \leq_p (y, y)$, since $x_1 \leq y \leq x_2$. If A extends O, then by monotonicity of A, we find that $A(x_1, x_2) \leq_p (O(y), O(y))$, therefore $O(y)$ is approximated by $A(x_1, x_2)$. This holds for arbitrary $(x_1, x_2) \in \boldsymbol{\mathcal{A}_s}$ and $y \in \boldsymbol{\mathcal{E}}$, therefore A approximates O. □

Note that AFT allows for multiple approximators of a fixed operator O. On top of that, AFT induces relationships between fixpoints of these different approximators. Specifically it defines an order on the approximators.

Definition 2.3.12 (Precision-Order on Approximators)**.** *An approximator A_1 is less precise than an approximator A_2, denoted by $A_1 \leq_p A_2$, if for every approximant (x_1, x_2), $A_1(x_1, x_2) \leq_p A_2(x_1, x_2)$.*

A $\leq_p$-maximal approximator U_O approximating an operator O is called an *ultimate approximator*[2] of O. It is easily verified that all ultimate approximators coincide for the consistent pairs, in particular, for every $x_1, x_2 \in \boldsymbol{\mathcal{E}}$, if $x_1 \leq x_2$, then $U_O(x_1, x_2) = (\mathrm{glb}_{\boldsymbol{\mathcal{C}}}\{O(y) \mid y \in [x_1, x_2]\}, \mathrm{lub}_{\boldsymbol{\mathcal{C}}}\{O(y) \mid y \in [x_1, x_2]\})$, where $[x_1, x_2] = \{y \in \boldsymbol{\mathcal{C}} \mid x_1 \leq y \leq x_2\}$. The following example describes an ultimate approximator of O_2.

Example 2.3.13 (Example 2.3.1.b, contd.)**.** *Below is one possible approximator A for the operator O_2. It is easy to check that A is $\leq_p$-monotonic and symmetric.*



With an approximator A, several types of fixpoints are definable. Since A is $\leq_p$-monotonic on the approximation space $\boldsymbol{\mathcal{A}_s}$ which is a complete lattice, it has a least (precise) fixpoint, this is the *Kripke-Kleene fixpoint* (KK-fixpoint) of A, denoted by $KK(A)$, i.e., $KK(A) = \mathrm{lfp}(A)$. By Theorem 2.2.11, the KK-fixpoint of A can be obtained constructively via monotonic inductions of A, where each step refines the lower and upper bound by deriving 'certain knowledge', i.e., knowledge which is known to hold. Moreover, the exact fixpoints of A correspond to the fixpoints of O; these are the *supported fixpoints* of A (SUP-fixpoints) denoted by $SUP(A)$, i.e., $SUP(A) = \{x \in \boldsymbol{\mathcal{E}} \mid A(x, x) = (x, x)\}$.

If the KK-fixpoint is an exact pair, i.e., $KK(A) = (x, x)$ for some $x \in \boldsymbol{\mathcal{E}}$, then x is the only fixpoint of O. Otherwise, it approximates all fixpoints of O, including *self-supported* ones that are in general not minimal in $\boldsymbol{\mathcal{E}}$.

[2]When the operator is clear from the context, we will drop the subscript-notation.

Example 2.3.14 (Example 2.3.1.b, contd.)**.** *In the diagram above, it is easy to see that the aforementioned approximator has four fixpoints, i.e., the least and most precise approximating pairs* $(\emptyset, \{a, b\})$ *and* $(\{a, b\}, \emptyset)$*, and the exact approximating pairs* $(\{a\}, \{a\})$ *and* $(\{b\}, \{b\})$*.*

The KK-fixpoint $KK(A) = \text{lfp}(A)$ *can be obtained using a monotonic induction starting from the least precise approximating pair* $(\emptyset, \{a, b\})$*. Since this is a fixpoint of* A *it is impossible to refine this any further, i.e., it is the least fixpoint of* A*.*

Intuitively, a monotonic induction of A *for this application can be explained as follows: in the beginning we know nothing about what our team will look like, every possible team is treated as a possibility. Then, we can increase the lower bound by adding people to the team of whom we are sure will join the team in any possible case. Similarly, we can decrease the upper bound by restricting the possible teams to all those that exclude people whenever we are sure they would never join the team in any of the possible scenarios. In this case, for both Alice and Bob, there exist scenarios where they would not join the team. Hence, there is no 'certain positive knowledge', thus we cannot improve the lower bound. Similarly, for both Alice and Bob, scenarios exist that would cause them to join, hence we cannot improve the upper bound either.*

Clearly, the KK-fixpoint $KK(A) = \text{lfp}(A) = (\emptyset, \{a, b\})$ *approximates both fixpoints of* O*, i.e.,* $\{a\}$ *and* $\{b\}$*. Furthermore, as expected, the SUP-fixpoints are exactly the fixpoints of* O*, i.e.,* $SUP(A) = \{\{a\}, \{b\}\}$*.*

Other important fixpoints are the A-stable fixpoints (ST-fixpoints) and the well-founded fixpoint (WF-fixpoint). The A-stable and WF-fixpoint definitions contain mechanisms to reduce self-support. The notion of self-support is not easily explained in an algebraic setting but shows intuitively in Example 2.3.1. Taking a closer look at the problem description, it becomes clear that Bob will only be added to the team if Bob is in the team. In this sense the inclusion of Bob is self-supported. In contrast, the fixpoint where Alice is the only team-member is not self-supported.

To mitigate self-support and to obtain the ST-fixpoints and the WF-fixpoint, AFT offers two approaches. Perhaps the most natural one is that of a well-founded induction, which formalizes the construction process. The limit of such induction is the WF-fixpoint. Alternatively, AFT defines the WF-fixpoint non-constructively as the least fixpoint of an operator St_A derived from the approximator A. The two formalizations are equivalent, i.e., they derive the same semantics. While the second approach provides a clear practical algebraic algorithm to compute the fixpoint, the well-founded inductions provide us with a good informal explanation of the solution.

As we believe it is the most natural explanation of AFT, we start with well-founded inductions. A well-founded induction is a sequence of refinements of our knowledge. It mirrors the construction process. At each step, our knowledge (captured by the lower and upper bounds) gets more and more precise. We have two types of refinements. First, an *application refinement* corresponds to a refinement that is consistent with applying the approximator. For instance, in Example 2.3.1.a, regardless of the initial set of people with access, applying the access policy will always grant Alice access. As a result, we can safely derive that Alice is in the set of people with access. This type of refinement, based on the inevitability of the inclusion or exclusion, can be captured by application refinements.

Second, a *grounding refinement* corresponds to a refinement where we remove unfounded sets. In particular, a grounding refinement refines the upper bound while ensuring that the obtained approximant is still closed under the semantic operator. For example, in Example 2.3.1.b we know that if we remove Bob from the set of people that may have access to the file (by lowering the upper bound to $\{a\}$), then none of the remaining possible scenarios would grant Bob access. Hence, the approximant $(\emptyset, \{a\})$ is closed under O_2.

Definition 2.3.15 (Refinement)**.** *For some $A \in Appr(\boldsymbol{\mathcal{A}_s})$.*

- $(x', y') \in \boldsymbol{\mathcal{A}_s}$ *is an* application refinement *of* $(x, y) \in \boldsymbol{\mathcal{A}_s}$ *if*

$$(x, y) \leq_p (x', y') \leq_p A(x, y).$$

- $(x, y') \in \boldsymbol{\mathcal{A}_s}$ *is a* grounding refinement *of* $(x, y) \in \boldsymbol{\mathcal{A}_s}$ *if*

$$x \leq y' \leq y \text{ and } (x, y') \leq_p A(x, y').$$

$(x', y') \in \boldsymbol{\mathcal{A}_s}$ is a refinement *of $(x, y) \in \boldsymbol{\mathcal{A}_s}$ if it is an application refinement of (x, y) or it is a grounding refinement of (x, y). For every $(x, y) \in \boldsymbol{\mathcal{A}_s}$, we use the notations $R^a_{(x,y)}$, $R^g_{(x,y)}$ and $R_{(x,y)}$ to denote the set of its application, grounding and all refinements, respectively.*

A refinement corresponds to one step in the construction process. The construction process itself is thus formalized by a sequence of refinements, i.e., a well-founded induction.

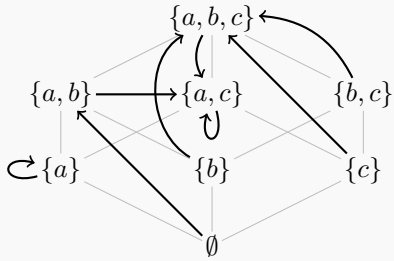
Definition 2.3.16 (Well-founded induction)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}_s})$. A* well-founded induction *of A is an increasing sequence $(x_i, y_i)_{i<\beta}$ with $(x_i, y_i) \in \boldsymbol{\mathcal{A}_s}$, satisfying*

- *(x_{i+1}, y_{i+1}) is a refinement of (x_i, y_i), for successor ordinals $i + 1 \leq \beta$,*
- *$(x_\lambda, y_\lambda) = \mathrm{lub}_{\boldsymbol{\mathcal{A}_s}}\{(x_i, y_i) \mid i < \lambda\}$, for limit ordinals $\lambda \leq \beta$.*

A well-founded induction $(x_i, y_i)_{i<\beta}$ of A is terminal *if there does not exist a strictly more precise refinement of its limit (x_β, y_β).*

Example 2.3.17 (Example 2.3.1.b, contd.)**.** *Remember the situation where owner grants access to Alice and Bob with the following access policy:*

- *Alice always blocks Bob's access.*
- *Bob always grants Charles access.*
- *Charles always grants himself access.*



Assume we use the ultimate approximator. We can formalize the construction process by the following well-founded induction:

$$(\emptyset, \{a, b, c\}) \to (\{a\}, \{a, b, c\}) \to (\{a\}, \{a\})$$

where the first refinement is an application refinement and the second is a grounding refinement. In the first step we derived that Alice will definitely be in the set. In the second step, we concluded that we can safely exclude Bob and Charles as, because Alice is in the set, Bob will be blocked and as a result, Bob and Charles form an unfounded set which we can delete.

Alternatively, the stable and well-founded fixpoint can be defined as a fixpoint of an operator, i.e., the stable revision operator.

Definition 2.3.18 (Stable Revision Operator)**.** *Let A be an approximator. The* stable revision operator *is given by $St_A : \boldsymbol{\mathcal{A}_s} \to \boldsymbol{\mathcal{A}_s}$ which maps any $(x_1, x_2) \in \boldsymbol{\mathcal{A}_s}$ to*

$$St_A(x_1, x_2) = (\mathrm{lfp}(A^1_{x_2}), \mathrm{lfp}(A^2_{x_1})),$$

where for every $y \in \boldsymbol{\mathcal{E}}$, $A^1_{x_2}(y)$ is the projection of $A(y, x_2)$ on its first component and $A^2_{x_1}(y)$ is the projection of $A(x_1, y)$ on its second component. To simplify notation, we will sometimes drop the subscript A when the approximator is clear from the context.

Since A is $\leq_p$-monotonic, it is easily derived that for any $x_2 \in \boldsymbol{\mathcal{E}}$, $A^1_{x_2}$ is $\leq$-monotonic on $\boldsymbol{\mathcal{E}}$, by definition of the precision order. Similarly, $A^2_{x_1}$ is $\leq$-monotonic on $\boldsymbol{\mathcal{E}}$ for every $x_1 \in \boldsymbol{\mathcal{E}}$. Consequently, by the Knaster-Tarski theorem, $\mathrm{lfp}(A^1_{x_2})$ and $\mathrm{lfp}(A^2_{x_1})$ exist, hence the stable operator is well-defined for any approximating pair (x_1, x_2).

Requiring A to be symmetric allows us to simplify the definition of the stable revision operator. In particular, since A is symmetric, the auxiliary operators $A^2_{x_1}$ and $A^1_{x_1}$ are equal. Hence, the stable revision operator $St_A(x_1, x_2)$ can be defined as $(\mathrm{lfp}(A^1_{x_2}), \mathrm{lfp}(A^1_{x_1}))$, reducing the amount of required auxiliary operators by half. Moreover, Denecker et al. [32] show that the symmetry of A entails that St_A preserves consistency.

Proposition 2.3.19 (Denecker et al. [32] Theorem 23.4)**.** *Let A be an approximator on $\boldsymbol{\mathcal{A}_s}$. The stable revision operator St_A preserves consistency, i.e., if (x_1, x_2) is consistent, then $St_A(x_1, x_2)$ is consistent as well.*

Theorem 2.3.20 (Denecker et al. [32] Proposition 20)**.** *Let A be an approximator on $\boldsymbol{\mathcal{A}_s}$. The stable revision operator St_A is a monotonic operator on $\boldsymbol{\mathcal{A}_s}$.*

By Proposition 2.3.3 $\boldsymbol{\mathcal{A}_s}$ is a complete lattice, therefore the stable revision operator St is a $\leq_p$-monotonic operator on a complete lattice. Thus, by the Knaster-Tarski theorem, it has a least (precise) fixpoint, i.e., the *Well-founded fixpoint* of A, denoted by $WF(A)$. This is exactly the limit of a terminal well-founded induction

Theorem 2.3.21 (Denecker and Vennekens [30] Theorem 3 [3])**.** *Let A be an approximator. If $(x_i, y_i)_{i<\beta}$ is a terminal well-founded induction for A, then $(x_i, y_i)_{i<\beta}$ converges to the well-founded fixpoint, i.e., the well-founded fixpoint of A can be constructed as the limit of any terminal well-founded induction of A.*

Example 2.3.22 (Example 2.3.1.b, contd.)**.** *Here we illustrate the other method to compute the well-founded fixpoint of Example 2.3.1.b, i.e., using monotonic inductions of the stable revision operator.*

Iteration 1. *The stable operator consists of two components. In particular, both components are least fixpoints of other (monotonic) operators, and can be constructed using monotonic induction of said operators.*

- *The lower bound is constructed as the least fixpoint of $A^1_{\{a,b\}}$:*

$$\emptyset \to A^1_{\{a,b\}}(\emptyset) = A^1(\emptyset, \{a, b\}) = \emptyset$$

[3] Denecker and Vennekens [30] formulated this theorem for gracefully-degrading approximators. As any symmetric approximator is gracefully-degrading, this includes all approximators considered here.

- *The upper bound is constructed as the least fixpoint of $A^2_\emptyset$:*

$$\emptyset \rightarrow A^2_\emptyset(\emptyset) = A^2(\emptyset, \emptyset) = \{a\} \rightarrow A^2_\emptyset(\{a\}) = A^2(\emptyset, \{a\}) = \{a\}$$

Thus after one iteration we obtain the approximant $(\emptyset, \{a\})$.

Iteration 2.

- *The new lower bound is now constructed as the least fixpoint of $A^1_{\{a\}}$:*

$$\emptyset \rightarrow A^1_{\{a\}}(\emptyset) = A^1(\emptyset, \{a\}) = \{a\}$$

- *The new upper bound is again constructed as the least fixpoint of $A^2_\emptyset$ since the lower bound remained unchanged, therefore, the new upper bound is again $\{a\}$.*

Thus after the second iteration we obtain the exact approximant $(\{a\}, \{a\})$, this is a fixpoint of the stable operator. Therefore, the WF-fixpoint is $(\{a\}, \{a\})$, moreover, $\{a\}$ is the unique A-stable fixpoint. As desired, the self-supported fixpoint $\{b\}$ was discarded. To link back to the original problem description, according to the WF-fixpoint, you would end up with Alice as your team-member.

Finally, the set of *A-stable fixpoints*, denoted by $ST(A)$, is defined as $\{x \in \boldsymbol{\mathcal{E}} \mid St_A(x, x) = (x, x)\}$, i.e., the exact fixpoints of St_A. As an alternative characterization, an A-stable fixpoint x is an element such that $x = \text{lfp}(A^1_x)$. Interestingly, this means that, while there is no constructive way to derive the stable fixpoints, there is a constructive way to test whether an exact element is a stable fixpoint.

We know that $KK(A)$ and $WF(A)$ are consistent and $KK(A) \leq_p WF(A)$. Moreover, every A-stable fixpoint x is approximated by the WF-fixpoint, and consequently also by the KK-fixpoint. Furthermore, if x is an A-stable fixpoint, then x is a minimal fixpoint of O.

Interestingly, if an approximator A is point-wise less precise than an approximator B, then $KK(A) \leq_p KK(B)$, $WF(A) \leq_p WF(B)$ and any stable fixpoint of A is a stable fixpoint of B. Thus, with increasing precision of the approximator, KK and WF fixpoints increase in precision, and the set of stable fixpoints grows.

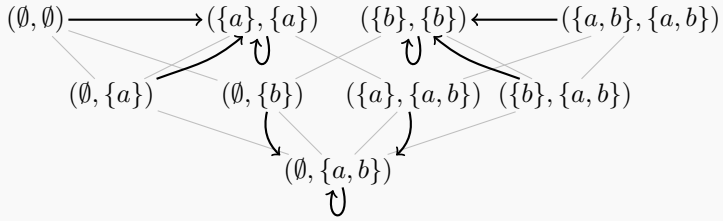
2.3.2 Interval AFT

Later, Denecker et al. [36] established consistent AFT where the approximation space is the subposet of $\langle \mathcal{P}^2, \leq_p \rangle$ induced by the set of consistent pairs $\mathcal{P}^c$. In general, consistent AFT is more natural while it still provides sufficient expressiveness. In this approximation space, every approximating pair corresponds to an interval, we will use the term *Interval AFT* (iAFT) to distinguish this version of AFT from the others.

Proposition 2.3.23. *If $\mathcal{P}$ is a complete lattice, then $\langle \mathcal{P}^c, \leq_p \rangle$ is a cpo.*

It was proven that every consistent approximator A is expandable to $\mathcal{P}^2$ approximators, and that each such an expansion has the same KK, WF and stable models as A. Vice versa, it is trivial to see that each approximator on $\mathcal{P}^2$ gives rise to an operator on $\mathcal{P}^c$ when restricted to the consistent pairs. For example, every $\leq_p$-maximal approximator on $\mathcal{P}^2$ leads to the unique $\mathcal{P}^c$-ultimate approximator U_O where $U_O(x_1, x_2) = (\text{glb}_{\mathcal{C}}\{O(y) \mid y \in [x_1, x_2]\}, \text{lub}_{\mathcal{C}}\{O(y) \mid y \in [x_1, x_2]\})$.

Example 2.3.24 (Example 2.3.1.b, contd.)**.** *Below is the ultimate approximator U for the operator O_2 on $\mathcal{P}^c$.*



The ideas and foundations of both versions of AFT mostly mirror each other. Therefore, here we only look at the differences between iAFT and sAFT, we do not repeat definitions and ideas that can be trivially reformulated for this space. In particular, there is one essential property from sAFT that is not trivially translated to iAFT, i.e., the monotonicity of the stable revision operator St.

Whereas in sAFT, St is $\leq_p$-monotonic on the entire approximation space, this is in general not true for consistent AFT. However, for an interesting subspace of the approximation space, the operator is internal and $\leq_p$-monotonic. In particular, we are interested in the A-reliable and A-prudent pairs.

Definition 2.3.25 (A-Reliable Approximant)**.** *Given an approximator A, an approximant (x_1, x_2) is* A-reliable *if $(x_1, x_2) \leq_p A(x_1, x_2)$.*

Definition 2.3.26 (A-Prudent Approximant)**.** *Given an approximator A, an approximant (x_1, x_2) is* A-prudent *if $x_1 \leq \text{lfp}(A^1_{x_2})$.*

The following theorem summarizes an important result for the subspace of A-reliable and A-prudent elements of the consistent approximation space.

Theorem 2.3.27 (Denecker et al. [36] Theorem 3.11)**.** *Let $\boldsymbol{\mathcal{A}} = \langle \mathcal{P}^c, \leq_p \rangle$ be a consistent approximation space with approximator A. The set of A-reliable and A-prudent elements of $\boldsymbol{\mathcal{A}}$ is a cpo under the precision order $\leq_p$ with least element $(\bot, \top)$. The stable revision operator is a well-defined, increasing and monotonic operator in this poset.*

In summary, in both versions of AFT, the stable revision operator St is a $\leq_p$-monotonic operator on a cpo. Therefore, it has a least (precise) fixpoint, i.e., the *Well-founded fixpoint* of A, denoted by $WF(A)$. The set of *A-stable fixpoints*, denoted by $ST(A)$, is defined as $\{x \in \boldsymbol{\mathcal{P}} \mid St(x, x) = (x, x)\}$, i.e., the exact fixpoints of St. As an alternative characterization, any A-stable fixpoint x_2 is an element such that $x_2 = \mathrm{lfp}(A^1_{x_2})$. In case of consistent AFT, the domain of $A^1_{x_2}$ is $[\bot, x_2]$ and that of $A^2_{x_1}$ is $[x_1, \top]$ while the range of both operators is $\boldsymbol{\mathcal{P}}$. So, it is possible that the iterated least fixpoint construction of these operators terminates in a point outside the operator domain in which case the operator has neither a fixpoint nor a least fixpoint. To accommodate, we call x_2 a stable fixpoint of A if the least fixpoint of $A^1_{x_2}$ exists and is equal to x_2. Other than these notions, everything else is completely analogous to sAFT.

2.3.3 Other AFT Versions

Beyond its original formulation, several variants of AFT have been developed to address the needs of different logical settings. For example, Charalambidis et al. [22] extended AFT to higher-order logic, thereby enabling the treatment of definitions over function and predicate variables, a necessity for more expressive forms of knowledge representation. This version of AFT helped inspire the more general framework for gAFT which we introduce in Chapter 5. Instead of defining the lower bounds and upper bounds in terms of the exact space, they allow for more flexible choices of approximants. In fact, the complete lattices $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$, respectively representing the lower and upper bounds, may differ. They are, however, strongly related. Specifically, they satisfy

- the *Interlattice Lub Property*: Let $u \in \boldsymbol{\mathcal{U}}$ and let $S \subseteq \boldsymbol{\mathcal{L}}$. If for every $l \in S$, $l \preceq u$, then $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(S) \preceq u$.
- the *Interlattice Glb Property*: Let $l \in \boldsymbol{\mathcal{L}}$ and let $S \subseteq \boldsymbol{\mathcal{U}}$. If for every $u \in S$, $l \preceq u$, then $l \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S)$.

We will refer to this version as hAFT.

Additionally, Heyninck et al. [50] studied AFT in the context of non-deterministic constructs, for example in the context of disjunctive logic programs, where multiple alternative conclusions can be derived from the same body. In this work, they show that AFT can capture and unify different semantics for disjunctive reasoning. Interestingly, this work not only changes the approximation space compared to iAFT and sAFT, but it also defines alternative versions of the stable revision operator and of an approximator. We will use dAFT to denote this version of AFT.

Other research has investigated inconsistent extensions of AFT (e.g., Bi et al. [8]) motivated by the observation that reasoning systems must sometimes cope with contradictory information to derive the fixpoint.

Together, these variations illustrate the flexibility of AFT as a unifying semantic framework, adaptable to a wide range of KR formalisms while preserving its core principles of approximation and fixpoint construction.

Chapter 3

Towards a Unifying View on Monotonic Constructive Definitions

3.1 Introduction

Within the landscape of constructive knowledge, few concepts are as intriguing and fundamental as inductive or recursive definitions.[1] Students become familiar with them through examples such as the transitive closure of a binary relation, the Fibonacci function or the satisfaction relation of propositional or predicate logic.

Common to such definitions is that they define a concept by describing how to construct it through iterated application of rules. This construction process is often called the *induction process*. For definitions of sets, the defined set is often explained non-constructively as the least set satisfying the rules; the constructive and non-constructive ways are known to be equivalent. While usually, it is not formally explained what inductive definitions mean, students apparently learn to understand them and reason with them. Brouwer [16], the famous constructionist, observed that many fundamental objects in mathematics were defined by describing how to *construct* them and that in understanding these constructions, we rely on our basic cognitive skills for *temporal*

[1] Is there a difference between *inductive* and *recursive* definition? According to some there is, according to others not. In this chapter, we propose a way to distinguish *inductive* and *recursive* definitions that is sensible and seems to match with intuitions of some.

reasoning.[2] It was later argued that also our skills for *causal reasoning* play a role here [26, 28, 31]: the hypothesis is that our understanding and reasoning capabilities for inductive definitions stem from our understanding of the induction process as a causal process, idealized and generalized to an (often infinite) universe of mathematical objects. This suggests a strong, but not well-known link between mathematics and common sense knowledge. While recently a lot of effort in the domain of large language models has resulted in surprisingly good commonsense reasoners, it is well-known that these are not reliable enough for sensitive applications where exactness and correctness are crucial. For these applications a logic-based approach that includes different constructive definitions is desired. Therefore, the study of the principle of inductive definition is a worthy topic in Knowledge Representation (KR).

It is clear that the concept of constructive definitions plays an important role in mathematics and foundations of computer science. We claim it also plays an important role in KR, at the meta-level (e.g., in the many inductive definitions used to define syntax and semantics for logics in KR), and at the object-level, since definitions constitute an important, common and precise form of human knowledge. In an important class of applications, the definition is constructive, in which case it is often not expressible in first-order logic (FO), yielding a second reason for studying constructive definitions [26]. A third reason is the intuition of some researchers that inductive and recursive definitions form the declarative understanding of two well-known declarative programming paradigms, logic and functional programming [33, 51]. Finally, due to the close connection between constructive definitions and causal information, studying constructive definitions is useful for expressing common sense causal knowledge [26, 28].

There exists extensive research on inductive definitions [82, 42, 61, 63, 4]. Also (co-) recursive definitions have received a lot of attention in relation to functional programming languages [73, 80, 75], as well as in *domain theory* where the functions, and the domain they are defined on are defined simultaneously [78, 1]. While there is a high level of familiarity with certain types of constructive definitions, in the current state of the art, fairly little interaction between research on different types of definitions seems to exist, resulting in a lack of deep understanding of common principles and applications. Many researchers seem aware that their theories only cover part of the topic. Already a long time ago, Moschovakis [63] explained how Kleene [54] in early papers had consciously studied constructive definitions[3] but *explicitly had drawn back from studying all of them*. Another complicating factor has been that inductive/recursive definitions have often been studied from a recursion-theoretic point of view, as programs to compute truth or function values, rather than as plain definitions

[2]While we follow Brouwer in his views on the nature and importance of constructive definitions, we use standard mathematics and set theory (also in this work) whenever suitable.

[3]In his work, Kleene used the term *inductive definitions* to denote the overarching class which we call constructive definitions.

of a concept.

This work contributes to the study of monotonic constructive definitions by introducing some key concepts as the foundations for a unifying set-theoretic framework. This offers the key insight that all these different types are instances of the same basic constructive principles. This has important practical implications. Firstly, it entails that research on a particular type of definition might transcend its class and actually be applicable to all constructive definitions. E.g., *non-monotonic* inductive definitions have been researched algebraically [29], but non-monotonic recursive definitions remain uncharted territory. Our correspondence suggests a way to generalize the study of non-monotonic definitions to other types of constructive definitions. Secondly, we believe this framework will be instrumental to integrate different types of definitions in a single knowledge representation language.

We see that in proof assistants based on type theory, such as Coq, Agda, or Lean, the principles of induction and co-induction on the one hand and recursion and co-recursion on the other, are all well-integrated in the proof systems [64, 9]. However, they are still treated as very distinct categories of definitions and very little attention is spent on their similar underlying principles. Moreover, while proof assistants are very useful tools in the domain of formal verification, they lack flexibility and versatility when it comes to knowledge representation. A good knowledge representation language should allow the knowledge to be applied in more applications other than for proof verification. We believe this warrants a deeper understanding of the underlying principles and informal semantics of constructive definitions; an understanding that goes beyond algorithms and inference rules as in proof theoretical settings.

Our main contribution is to show how a whole range of examples from different areas can be reduced to instantiations of the same fundamental principles, using standard set-theoretic constructions. First, we recall the principle of (monotonic) inductive definition and its formalization in fixpoint theory, which will involve a *semantic operator* on a so-called *construction space*, which is often richer than the *object space*, in which the *defined object* naturally lives. We then analyze examples found in a wide range of areas. In each example, we describe the object space, the construction space, the monotonic semantic operator and the defined entity. We will see how the construction space can be used as the key factor to distinguish between classes of definitions from different research areas. We focus mostly on (co)inductive definitions of sets and (co)recursive definitions of functions, but also briefly discuss some more complex types of constructive definitions.

This chapter contains and extends the published work in Vanbesien et al. [87], adapted to fit in the current formalization. Section 3.4 concerns unpublished work.

3.2 Algebraic Formalization

This section introduces the algebraic foundations needed for a deeper analysis of constructive definitions, drawing on order and fixpoint theory and illustrating the concepts with a detailed example. A constructive definition for a concept $\mathcal{D}$ is (formalized as) an operator $O : \boldsymbol{\mathcal{C}} \to \boldsymbol{\mathcal{C}}$ on a cpo $\boldsymbol{\mathcal{C}}$. It defines the object D representing $\mathcal{D}$ by describing how to construct it.

Following Aczel [4], this allows for a first, algebraic formalization of constructive definitions. If O is monotonic, then such construction, normally called the *induction process*, corresponds to the sequence $(O^i)_{i\geq 0}$, i.e., the fastest monotonic induction of O. The defined object D is the limit of this sequence. This limit can be obtained by construction, and as a direct consequence of Theorem 2.2.9, it can also be characterized non-constructively, as the least (pre-)fixpoint of O, yielding the duality between the constructive and non-constructive view on inductive definitions.

This abstract formalization of monotonic constructive definitions and the induction process is natural, abstract, elegant, powerful and is widely adopted in mathematical logic studies. Let us illustrate it on a prototypical example. To streamline the presentation of various examples, we initially present a constructive definition as a set $\mathcal{R}$ of rules[4] which resemble the style used in logic programming, as well as in functional programming. We believe this will lead to an improved understanding of our examples. Moreover, it gives an idea of how constructive definitions in natural language can be formalized, which is essential when developing knowledge representation languages that include them.

Example 3.2.1 (Transitive closure)**.** *Let $\mathcal{G} = (V, E)$ be a directed graph. The set F of edges of the* transitive closure *$\mathcal{T} = (V, F)$ of $\mathcal{G}$ is defined inductively:*

- *$(x, y) \in F$ if $(x, y) \in E$;*
- *$(x, y) \in F$ if there exists a vertex z such that $(x, z) \in F$ and $(z, y) \in F$.*

The set of rules $\mathcal{R}_F$ defining $\mathcal{T} = (V, F)$ is as follows.

$$\left\lfloor \begin{array}{l} \forall x \forall y : F(x, y) \leftarrow E(x, y). \\ \forall x \forall y : F(x, y) \leftarrow \exists z : F(x, z) \wedge F(z, y). \end{array} \right\rfloor$$

[4] We use different brackets to indicate the kind of definition: inductive and recursive definitions will be enclosed in floor-brackets $\lfloor \mathcal{R} \rfloor$, coinductive and corecursive definitions in ceil-brackets $\lceil \mathcal{R} \rceil$, and any other kind of constructive definition in curly brackets $\{\mathcal{R}\}$.

We expect this definition to construct a set $F \subseteq V^2$ of edges. Hence, we consider the cpo $\boldsymbol{\mathcal{C}_F} = \langle 2^{V^2}, \subseteq \rangle$, with the power set of V^2 as the underlying set equipped with the subset-order. Moreover, the rules in $\mathcal{R}_F$ suggest an operator $O_F \colon \boldsymbol{\mathcal{C}_F} \to \boldsymbol{\mathcal{C}_F}$ showcasing rule application, by mapping a set $S \in \boldsymbol{\mathcal{C}_F}$ to

$$O_F(S) = E \cup \{(x, y) \mid (x, z), (z, y) \in S \text{ for some } z \in V\}.$$

It is not hard to prove that O_F is monotonic and it is easy to see that its least fixpoint is the set F of edges of the transitive closure of $\mathcal{G}$.

Proposition 3.2.2. *O_F is a monotonic operator.*

Proof. Let $S_1 \subseteq S_2$ be two subsets of V^2. We have to show that $O_F(S_1) \subseteq O_F(S_2)$. By definition of O_F, we have that $O_F(S_1) = E \cup \{(x, y) \mid \exists z : (x, z) \in S_1 \wedge (z, y) \in S_1\}$. Let $(x, y) \in O_F(S_1)$. If $(x, y) \in E$, then $(x, y) \in O_F(S_2)$. If $(x, y) \notin E$, then there exists $z \in V$ such that $(x, z), (z, y) \in S_1$. Since $S_1 \subseteq S_2$, $(x, y) \in O_F(S_2)$, as desired. □

In other words, O_F is an algebraic formalization of the constructive definition of F. The defined set F can be characterized non-constructively as the least fixpoint of O_F, or constructively as the limit of the induction process, i.e., the sequence built by iterative application of O_F starting from the empty set.

Denecker and Vennekens [31] remarked that given a set of informal rules such as in Example 3.2.1, we most likely picture the induction process as a sequence of applications of rule instances, rather than iterations of O_F. In this view of the induction process, the elementary step is the application of a rule instance (or perhaps more generally, the application of a set of rule instances). This natural view of the induction process gave rise to the more general notion of a monotonic induction. It identifies the *rule* as the modular unit of the definition and its induction process. This modularity is abstracted away when formalizing the definition as an operator O. Moreover, this view gives rise to a highly non-deterministic notion of induction process, since rules can be applied in different orders. This non-determinism is of great practical use when reasoning on the definition, since it allows us to *steer* the induction process towards a particular goal, e.g., towards computing whether a specific pair (a, b) belongs to F. Crucially, all induction processes are confluent, i.e., they converge to the same limit.

In Fig. 3.1, the start of two such induction processes for Example 3.2.1 are visualized. In the top sequence $(F_0, \ldots, F_3)$, all applicable rules are applied at every step of the construction, making it the fastest process. This corresponds to $(O_F^0, \ldots, O_F^3)$, the first four iterations of the operator O_F. In the bottom sequence, a slower induction process $(F'_0, \ldots, F'_3)$ is shown, one that first applies all instances of the base rule, then a single instance of the transitivity rule per iteration.

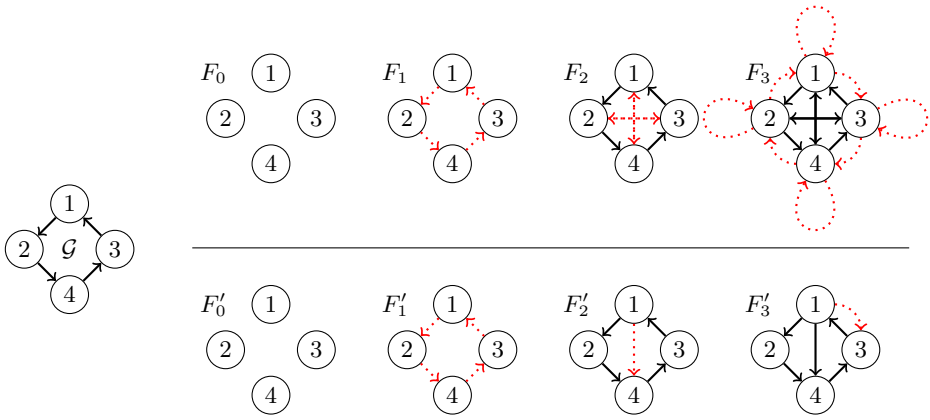


Figure 3.1: A graph $\mathcal{G}$ (left) and the start of two monotonic inductions of the definition of its transitive closure (right). Dotted red arrows indicate newly derived edges at each state.

In the next section, we present several examples of constructive definitions. While they originate from very different fields, they can be presented in a uniform way using the following mathematical objects:

- A mathematical object D corresponding to the concept defined by the constructive definition. We call D the *defined object*.
- A set $\boldsymbol{\mathcal{O}}$ where D 'lives', i.e., a space in which the object exists or can be defined. This should be naturally identified by the specifications in the constructive definition. We call $\boldsymbol{\mathcal{O}}$ the *object space*.
- A cpo $\boldsymbol{\mathcal{C}} = \langle \mathcal{C}, \leq \rangle$ with an injection $\theta \colon \boldsymbol{\mathcal{O}} \hookrightarrow 2^{\mathcal{C}} \setminus \{\emptyset\}$ (rather than $\boldsymbol{\mathcal{O}} \hookrightarrow \mathcal{C}$, as will be justified later) such that for all $d_1, d_2 \in \boldsymbol{\mathcal{O}}$ with $d_1 \neq d_2$, $\theta(d_1) \cap \theta(d_2) = \emptyset$, i.e., different elements of the object space are mapped to disjoint sets. We call $\boldsymbol{\mathcal{C}}$ the *construction space*.
- An operator $O \colon \boldsymbol{\mathcal{C}} \to \boldsymbol{\mathcal{C}}$ on the construction space, of which the least fixpoint coincides with the defined object D: $\mathrm{lfp}(O) \in \theta(D)$. We call O the *semantic operator*.

The elements of the construction space can be seen as representing partial or lower-bound information about the elements of the object space. Each element $c \in \boldsymbol{\mathcal{C}}$ corresponds to at most one object $d \in \boldsymbol{\mathcal{O}}$, namely the unique d such that $c \in \theta(d)$, if

such a d exists. Conversely, a single object d may correspond to multiple construction elements $c \in \mathcal{C}$, reflecting the fact that an object can sometimes be obtained through different construction processes resulting in different representations of that same object. For instance, in Example 3.3.17, the exact space consists of a set of functions, whereas the construction space contains extensions of these functions. Different semantic operators may construct different extensions corresponding to the same function. Each extension $f^* \in \mathcal{C}$ of a function $f \in \mathcal{O}$ therefore must serve as a representation of f, that is, $f^* \in \theta(f)$.

Complementary, the mapping θ defines a surjective partial function $\pi \colon \mathcal{C} \twoheadrightarrow \mathcal{O}$ such that for $c \in \mathcal{C}$, $\pi(c)$ is defined and equal to d iff $c \in \theta(d)$. In practice, π is used to project away the extra construction details and retrieve the corresponding value in the object space, for example by applying it to the least fixpoint of the operator: $D = \pi(\mathrm{lfp}(O))$. Often but not always, the object and the construction space are the same and π is the identity function. E.g., in Example 3.2.1, the defined object is the set F of edges of the transitive closure, the object space $\mathcal{O}_F$ is the set 2^{V^2} of all sets of possible edges; the construction space is $\mathcal{C}_F = \langle 2^{V^2}, \subseteq \rangle$; the semantic operator is O_F.

3.3 Different Flavors of Constructive Definitions

In this section, we instantiate the earlier introduced framework for a range of constructive definitions coming from different areas. We bring them together to show that indeed, in different fields, the design of the object and construction space is the key point. Once this choice is made explicit, typically the definition of the operator follows straightforwardly, and the defined object is constructed by the fixpoint theory. The final step may be to project the fixpoint from the construction space to an element of the object space, i.e., the defined object, using π. In the majority of the proposed examples, this is not needed, since the injection θ just sends an exact element $d \in \mathcal{O}$ to the singleton $\{d\} \in 2^{\mathcal{C}}$. However, Example 3.3.17 illustrates where the projection π plays a role.

3.3.1 (Co-)inductive Definitions of Sets

Inductive definitions are ubiquitous in mathematical texts. Concepts such as the transitive closure, the natural numbers, ordinals, and formulas in logic, are usually defined inductively [5, 4]. On the other hand, many common infinite data-types such as infinite streams, infinite trees and co-terms, are typically defined co-inductively [56].

In general, these definitions define sets of elements of a certain type $\mathcal{T}$, given by the context. Naturally, the object space then consists of all sets of elements of $\mathcal{T}$, i.e., it is

$2^{\mathcal{T}}$. Intuitively, the construction process associated with inductive definitions gradually grows the defined set, starting from the empty set. In contrast, the construction process for co-inductive definitions puts stronger restrictions on the defined set in every step, resulting in a gradually shrinking set. In both cases the power set contains all elements necessary for the construction, since we are only adding or removing elements from a subset of $\mathcal{T}$. By endowing the object space with the subset order $\subseteq$ and the superset order $\supseteq$, we capture the respective behaviors of growing and shrinking associated with induction and co-induction. For inductive definitions we obtain the power set lattice $\langle 2^{\mathcal{T}}, \subseteq \rangle$ as a construction space. This is a complete lattice and thus a cpo. The same holds for the construction space $\langle 2^{\mathcal{T}}, \supseteq \rangle$.

The inverted order explains why co-induction is often thought of as the greatest fixpoint with respect to $\subseteq$, instead of as the least fixpoint with respect to $\supseteq$.

Terminating and Non-Terminating Nodes

Example 3.3.1 (Terminating node)**.** *Let $\mathcal{G} = (V, E)$ be a finite directed graph. A node is* terminating (in $\mathcal{G}$) *if all of its successors in $\mathcal{G}$ are terminating.*

In formal notation, the set T of terminating nodes is defined inductively by the rule

$$\lfloor\ \forall x \in V : T(x) \leftarrow (\forall y \in V : E(x, y) \Rightarrow T(y))\ .\ \rfloor$$

where '$\Rightarrow$' denotes material implication.

If $\mathcal{G}$ is a graph, the set T is precisely the set of nodes from which no infinite path starts. Clearly, the object space $\boldsymbol{\mathcal{O}_T}$ is thus the set of all sets of vertices, i.e., 2^V, while for the construction space we use $\langle 2^V, \subseteq \rangle$. As before, the semantic operator $O_T : \boldsymbol{\mathcal{C}_T} \to \boldsymbol{\mathcal{C}_T}$ simply evaluates rule bodies. i.e., it maps a set S of vertices to $O_T(S)$ with

$$O_T(S) := \{x \mid y \in S \text{ for every } y \text{ where } (x, y) \in E\}$$

where E is given. It is easy to see that the i-th iteration of the operator adds all terminating nodes of which the longest outgoing path is of length $i - 1$.

Proposition 3.3.2. *The operator O_T is monotonic.*

Proof. Let $S_1 \subseteq S_2$ be two subsets of V. We have to show that $O_T(S_1) \subseteq O_T(S_2)$. By definition of O_T, we have $O_T(S_1) = \{x \mid y \in S \text{ for every } y \text{ where } (x, y) \in E\}$. Let $x \in O_T(S_1)$. There must exist an $y \in S_1$ such that $(x, y) \in E$. As $S_1 \subseteq S_2$, this entails $y \in S_2$. Hence, $x \in O_T(S_2)$ as desired. □

Fig. 3.2 visualizes the fastest monotonic induction for an example graph, obtained by iterating the operator. The induction process gradually grows the set of terminating

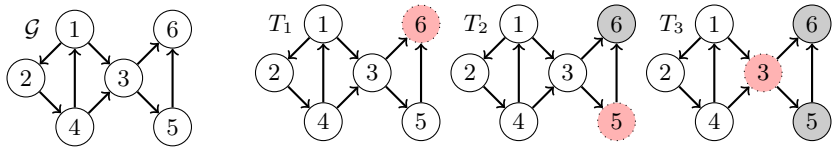


Figure 3.2: A graph (left) and a visualization of a monotonic induction for the definition of its set of terminating node. The nodes derived to be in the set are shaded, the newly derived nodes at each step are red and have a dotted edge.

nodes starting from the empty set by adding all elements supported by a rule. In contrast to the previous example, there is no separate rule for 'base-cases' that can be derived during the first iteration. Instead, this rule is encapsulated within the inductive rule. However, the base case is revealed when we simplify the inductive rule using $T_0 = \emptyset$, resulting in $\forall x \in V : T(x) \leftarrow \forall y \in V : \neg E(x, y)$. As such, during the first iteration only nodes with no outgoing edges can be added to the defined set. Later, more and more terminating nodes are identified until a fixpoint is reached. In general, the fastest monotonic induction for this definition is represented by the sequence

$$\emptyset = T_0 \subseteq T_1 \subseteq \ldots \subseteq T_n \subseteq \ldots \subseteq T$$

with $T_i = \{x \in V \mid \text{there is no path in } \mathcal{G} \text{ with length } i \text{ that starts from } x\}$. The limit of this sequence is exactly the set of terminating nodes we expect.

Now let us take a look at the definition of its set complement in V, i.e., the set of non-terminating nodes N.

Example 3.3.3 (Non-terminating node)**.** *Let $\mathcal{G} = (V, E)$ be a finite directed graph. The* set of non-terminating nodes N *is co-inductively defined as follows: a node is non-terminating if it has at least one child that is non-terminating.*

Formally, the set N of non-terminating nodes (i.e., nodes with an infinite outgoing path) is defined co-inductively by the rule:

$$\lceil \ \forall x \in V : N(x) \leftarrow \exists y \in V : E(x, y) \wedge N(y). \ \rceil$$

The object space is again 2^V. In contrast to its complement, the set of non-terminating nodes is defined co-inductively. This suggests using the construction space $\boldsymbol{\mathcal{C}_N} = \langle 2^V, \supseteq \rangle$. The semantic operator $O_N : \boldsymbol{\mathcal{C}_N} \to \boldsymbol{\mathcal{C}_N}$ now maps a set of vertices S to the set $O_N(S)$ such that:

$$O_N(S) := \{x \mid (x, y) \in E \text{ for some } y \in S\}.$$

Proposition 3.3.4. *The operator O_N is monotonic.*

Proof. Analogous to Proposition 3.3.2. □

In general, the fastest (co-)induction is given by the sequence

$$V = N_0 \supseteq N_1 \supseteq \ldots \supseteq N_n \supseteq \ldots \supseteq N$$

where each N_i is given by $V \setminus T_i = \{x \in V \mid$ there exists a path in $\mathcal{G}$ with length i that starts from $x\}$. Notice the symmetry with the fastest induction of the previous example. Simply changing the shaded nodes in Fig. 3.2 to non-shaded nodes and vice versa, would result in a perfect example for the current definition.

In general, given a monotonic inductively defined object D it is possible to define the set complement D^c co-inductively by negating the defining condition, and vice versa. The fastest induction processes for these definitions are always symmetric.

Finite and Infinite Lists of Primes

Next, let us consider the domain of (finite or infinite) lists of natural numbers. The set of all such lists is denoted by $List$. We use a well-known notation for lists where Nil represents the empty list and $[x \mid y]$ represents the list starting with $x \in \mathbb{N}$ (often referred to as the *head*) followed by the list y (often referred to as the *tail* of the list).

Example 3.3.5 (Prime array)**.** *The set PA of all* prime arrays *is defined inductively:*

- $Nil \in PA$.
- *If x is a prime number and $y \in PA$, then $[x \mid y] \in PA$.*

This is a monotonic inductive definition, formally represented by the set of rules

$$\left\lfloor \begin{array}{l} \forall y \in List : PA(y) \leftarrow y = Nil. \\ \forall x \in \mathbb{N}, \forall y \in List : PA([x \mid y]) \leftarrow P(x) \wedge PA(y). \end{array} \right\rfloor$$

with P the set of prime numbers.

The object space $\boldsymbol{\mathcal{O}_{List}}$ is the power set 2^{List}. As construction space we then have $\boldsymbol{\mathcal{C}_{PA}} = \langle 2^{List}, \subseteq \rangle$. The semantic operator for this example is $O_{PA} : \boldsymbol{\mathcal{C}_{PA}} \rightarrow \boldsymbol{\mathcal{C}_{PA}}$, defined by mapping a set of lists $S \subseteq \boldsymbol{\mathcal{C}_{PA}}$ to

$$O_{PA}(S) := \{l \mid l = Nil \text{ or } l = [x \mid y] \text{ for some } x \in P, y \in S\}$$

Proposition 3.3.6. *The operator O_{PA} is monotonic.*

Proof. Let $S_1 \subseteq S_2$ be two subsets of $List$, and let $l \in O_{PA}(S_1)$. Either $l = Nil$, in which case $x \in O_{PA}(S_2)$, or there exist $x \in P$ and $y \in S_1$ such that $l = [x \mid y]$. Since $S_1 \subseteq S_2$, we have $y \in S_2$, which implies that $x \in O_{PA}(S_2)$ also for the latter case. □

At step i of the fastest induction process all lists of length $i-1$ of primes are added to the defined set. This means that during the first iteration only the empty list Nil is derived to be a finite list of primes. In the subsequent step every list $[n \mid Nil]$ with n a prime number is also added to the set, and so on. This corresponds to the sequence

$$\emptyset = PA_0 \subseteq PA_1 \subseteq \ldots \subseteq PA,$$

with $PA_i = \bigcup_{m<i}\{[n_0, \ldots, n_m] \mid n_0, \ldots, n_m \in P\}$, the set of lists of primes with length at most i. Thus, the defined set of prime arrays consists of all finite lists containing only prime numbers. Interestingly, the same set of rules gives rise to a sensible *co-inductive* definition.

Example 3.3.7 (Prime lists)**.** *The set PL of all* prime lists *is defined co-inductively:*

- $Nil \in PL$.
- $[x \mid y] \in PL$, *if x is a prime number and $y \in PL$.*

As suggested before, this definition corresponds to exactly the same formal set of rules as the previous example after replacing PA by PL

$$\left[\begin{array}{l} \forall y \in List : PL(y) \leftarrow y = Nil. \\ \forall x \in \mathbb{N}, \forall y \in List : PL([x \mid y]) \leftarrow PL(y) \wedge P(x). \end{array}\right]$$

Unsurprisingly, we consider the same object space 2^{List} as in Example 3.3.5, and the construction space with inverted order, namely $\langle 2^{List}, \supseteq \rangle$. Except for its signature, the inverted order does not influence the semantic operator O_{PL}, which equals O_{PA}.

Proposition 3.3.8. *The operator O_{PL} is monotonic.*

Proof. Clear by the definition of O_{PL} and Proposition 3.3.6. □

As before, the fastest induction process starts from the least element in the lattice which is now $List$ and gradually constructs larger elements until a fixpoint is reached. During the first step all non-empty lists for which the first element is not a prime are discarded.

At each subsequent step i, all lists with length greater than $i-1$ where the ith element is not a prime are discarded. This results in the sequence

$$List = PL_0 \supseteq PL_1 \supseteq \ldots \supseteq PL,$$

with $PL_i = \bigcup_{m<i}\{[n_0,\ldots,n_m] \mid n_0,\ldots,n_m \in P\} \cup \{[n_0,\ldots,n_i,\ldots] \mid n_0,\ldots,n_i \in P\}$ where $[n_0,\ldots,n_i,\ldots]$ denotes a list with length greater than $i-1$. Intuitively, this set corresponds to all lists l of natural numbers such that no non-primes occur within the first i elements of the list. Clearly, this sequence converges to the set of all finite and infinite lists of prime numbers.

A final adaptation of the list-example restricts the defined object to only the infinite lists of prime numbers.

Example 3.3.9 (Prime streams)**.** *The set PS of all* prime streams *is defined co-inductively:*

- $[x \mid y] \in PS$ *if x is a prime number and $y \in PS$.*

By excluding the case for the empty list Nil, we obtain only the infinite lists, i.e., the streams. The definition is formalized by the following co-inductive rule:

$$\left\lceil\ \forall x \in \mathbb{N}, \forall y \in List : PS([x \mid y]) \leftarrow PS(y) \wedge P(x).\ \right\rceil$$

We keep the same object space and construction space as in Example 3.3.7. Here, the difference lies with the semantic operator O_{PS} which maps a set of lists S to

$$O_{PS}(S) := \{[y \mid z] \mid z \in S, y \in P\}$$

Proposition 3.3.10. *The operator O_{PS} is monotonic.*

Proof. Let $S_1 \subseteq S_2$ be two subsets of $List$, and let $l \in O_{PS}(S_1)$, i.e., $l = [y,z]$ for some $y \in P$ and $z \in S_1$. Since $S_1 \subseteq S_2$, we have $y \in S_2$, which implies that $x \in O_{PS}(S_2)$, as desired. □

The fastest induction process for this definition starts from $List$, since by default everything belongs to the set. During the first step it will delete the empty list and all lists with a head a such that $a \notin P$. At each subsequent step i it will remove all lists for which the ith element either does not exist, or is not a prime number, giving us the sequence

$$List = PS_0 \supseteq PS_1 \supseteq \ldots \supseteq PS,$$

with $PS_i = \{[n_0,\ldots,n_i,\ldots] \mid \forall j < i, n_j \in P\}$, i.e., the set of all lists of length at least i of which the first i elements are primes. Note that interpreting this set of rules

inductively rather than coinductively will not be able to derive the inclusion of a single element, i.e., the defined object would be the empty set.

Company Control

Let us turn our attention to a different type of example that uses an aggregate expression, known as the "company controls" problem [53].

Example 3.3.11 (Company control-relation)**.** *Given a set C of companies, each of which owns a percentage of the shares of the other companies, the control-relation is defined inductively as follows: a company x controls another company y, if the sum of the shares of y owned by x or by companies controlled by x, is strictly more than half. In formal rule notation:*

$$\left\lfloor \forall x, y \in C : Cont(x, y) \leftarrow \left(\sum_{z \in Cont^x} Sh(z, y) \right) > 0.5. \right\rfloor$$

where $Sh\colon C^2 \to [0, 1]$ is a function that maps a pair of companies (x, y) to the fraction of shares of y owned by x and $Cont^x = \{x\} \cup \{u \mid Cont(x, u)\}$.

Under the (natural) assumption that $Sh(x, y) \geq 0$, this definition is monotonic. The more companies that are determined to be under control of a company x, the higher the fraction of shares controlled by x in any (other) company y. The object space is now given by the set of binary relations over C, i.e., 2^{C^2}, as the construction space we choose $\boldsymbol{\mathcal{C}_{Cont}} = \langle 2^{C^2}, \subseteq \rangle$. Once again, the semantic operator $O_{Cont} : \boldsymbol{\mathcal{C}_{Cont}} \to \boldsymbol{\mathcal{C}_{Cont}}$ results from rule application, i.e., it maps a binary relation R to

$$O_{Cont}(R) := \left\{ (x, y) \;\middle|\; \sum_{z \in \{x\} \cup \{u \mid (x,u) \in R\}} Sh(z, y) > 0.5 \right\}.$$

Proposition 3.3.12. *The operator O_{Cont} is monotonic.*

Proof. Let $S_1 \subseteq S_2$ be two subsets of C^2, and let $(x, y) \in O_{Cont}(S_1)$, i.e.,

$$\sum_{z \in \{x\} \cup \{u \mid (x,u) \in S_1\}} Sh(z, y) > 0.5.$$

Since $S_1 \subseteq S_2$, we also have the inclusion $\{x\} \cup \{u \mid (x, u) \in S_1\} \subseteq \{x\} \cup \{u \mid (x, u) \in S_2\}$. Since $Sh(z, w) \geq 0$ for all $(z, w) \in C^2$, we have

$$\sum_{z \in \{x\} \cup \{u \mid (x,u) \in S_2\}} Sh(z, y) > 0.5,$$

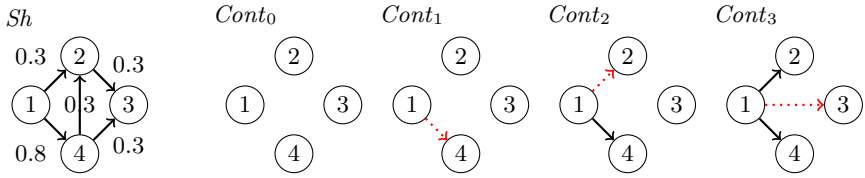


Figure 3.3: An edge (a, b) in the leftmost graph indicates that $Sh(a, b) > 0$ and its label shows the exact value of $Sh(a, b)$. The other graphs show an induction process. Newly derived company pairs (represented by edges) are indicated with dotted red lines. At first only the base case is added. Later, combined ownership is derived.

i.e., $(x, y) \in O_{Cont}(S_2)$. □

Fig. 3.3 visualizes the induction process for an example share-function Sh represented by a labeled directed graph. Coincidentally, the depicted induction is the only possible induction with strict increments since at every step exactly one rule is applicable.

3.3.2 (Co-)recursive Definitions of Functions

We now present another set of examples, this time regarding the definition of functions. Recursion and its dual co-recursion are extensively used as methods to define functions in a wide variety of mathematical and computer scientific fields [73, 80, 75]. Some well-known mathematical functions, like the factorial or the greatest common divisor, can be defined recursively, and (co)recursive definitions of functions are supported in most functional programming languages, like Haskell [38], Coq [76], Agda, and Scheme [39]: for instance, integer multiplication, functions returning the length or the maximum of a list, and depth-first tree traversal can easily be defined recursively, whereas breadth-first tree traversal can be implemented co-recursively.

For our formalization, the object space of (co)recursive definitions of functions is obtained in a natural way: if we want to define a function $f\colon X \to Y$, then the object space is the set of functions from X to Y, denoted by Y^X. Contrary to (co-)inductive definitions, here we cannot just choose the construction space to equal the object space. The main reason for this is that in intermediate steps of the construction process only partial functions have been constructed.

The Fibonacci Sequence

Example 3.3.13 (Fibonacci sequence)**.** *The Fibonacci sequence is viewed here as the function* $Fib\colon \mathbb{N} \to \mathbb{N}$*, where* $Fib(n)$ *is the* n*th Fibonacci number. Its recursive*

definition, in the formal notation, is the following:

$$\left\lfloor \begin{array}{l} Fib(0) := 0. \\ Fib(1) := 1. \\ \forall n \in \mathbb{N} : Fib(n+2) := Fib(n) + Fib(n+1). \end{array} \right\rfloor$$

Clearly, the object space is $\boldsymbol{\mathcal{O}_{Fib}} = \mathbb{N}^{\mathbb{N}}$. Moreover, we can get some insight into the construction process from the rules above. Note that the image of $n+2$ under Fib depends on the image of n and $n+1$. As long as the latter are not derived, it is impossible to determine $Fib(n+2)$. Hence, it is natural to think of Fib at an intermediate step of the construction as a partial function, defined on a subset S of $\mathbb{N}$. Equivalently, we can view such a partial function as a function from $\mathbb{N}$ to $\mathbb{N}_\perp := \mathbb{N} \cup \{\perp\}$, where $\perp$ denotes "undefined", sending natural numbers not in S to $\perp$.

In the following we present a possible choice for a suitable construction space and a monotonic operator for the recursive definition of Fib. Consider on $\mathbb{N}_\perp$ the partial order $\leq$ given by $\forall n \in \mathbb{N},\ \perp \leq n$.

Lemma 3.3.14. *$\langle \mathbb{N}_\perp, \leq \rangle$ is a cpo.*

Proof. Let $S \subseteq \mathbb{N}_\perp$ be a chain. By the definition of the order $\leq$, S has at most two elements. Hence, the $lub(S) = n$ where $\{n\} = S \cap \mathbb{N}$ if $S \cap \mathbb{N} \neq \emptyset$, or $\perp$ otherwise. □

Now we can define the construction space $\boldsymbol{\mathcal{C}_{Fib}}$. Let $\mathcal{C}_{Fib}$ be the space of functions from the natural numbers[5] to $\mathbb{N}_\perp$, and let $\leq_d$ be the point-wise partial order on $\mathcal{C}_{Fib}$. Let $\boldsymbol{\mathcal{C}_{Fib}} := \langle \mathcal{C}_{Fib}, \leq_d \rangle$.

Proposition 3.3.15. *$\boldsymbol{\mathcal{C}_{Fib}}$ is a cpo.*

Proof. Let $S \subseteq \mathcal{C}_{Fib}$ be a chain. Since the order $\leq_d$ is defined point-wise, for all $n \in \mathbb{N}$, $S_n := \{f(n) \mid f \in S\} \subseteq \mathbb{N}_\perp$ is a chain. By Lemma 3.3.14, for all $n \in \mathbb{N}$, there exists $\mathrm{lub}(S_n)$. It is easy to see that the function $F\colon \mathbb{N} \to \mathbb{N}_\perp$ defined by $F(n) := \mathrm{lub}(S_n)$ is the least upper bound of S. □

Notice that $\mathcal{C}_{Fib}$ contains all the intermediate functions needed for the steps of the recursion as presented above.

[5]The construction space $\boldsymbol{\mathcal{C}_{Fib}}$ contains functions with a domain different from the co-domain. We could consider functions from $\mathbb{N}_\perp$ to itself, but this is not needed as the recursion works perfectly without this enriched domain. On the other hand, Example 3.3.17 shows that such symmetry between domain and co-domain is sometimes necessary.

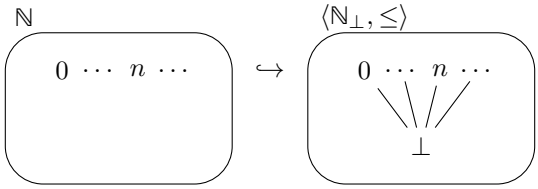


Figure 3.4: The signature of the elements in the construction space for the Fibonacci definition, i.e., on the left, the set of natural numbers $\mathbb{N}$, which are mapped to $\langle \mathbb{N}_\perp, \leq \rangle$ on the right.

Hence, it remains to define a monotonic operator on $\boldsymbol{\mathcal{C}_{Fib}}$. First, the sum $+\colon \mathbb{N}^2 \to \mathbb{N}$ can be extended to $\mathbb{N}_\perp \times \mathbb{N}_\perp$ by defining for each $n \in \mathbb{N}_\perp$, $\perp + n = n + \perp = \perp$. Then, the operator $O_{Fib}\colon \boldsymbol{\mathcal{C}_{Fib}} \to \boldsymbol{\mathcal{C}_{Fib}}$ is defined to map a function $f \in \boldsymbol{\mathcal{C}_{Fib}}$ to

$$O_{Fib}(f) := \begin{cases} n & \mapsto n \quad (\text{for } n \in \{0,1\}) \\ n+2 & \mapsto f(n+1) + f(n) \end{cases}$$

Proposition 3.3.16. *The operator O_{Fib} is monotonic.*

Proof. Let $f, g \in \boldsymbol{\mathcal{C}_{Fib}}$ such that $f \leq_d g$, i.e., for all $n \in \mathbb{N}$, $f(n) \leq g(n)$. Notice that we have

$$\begin{aligned} O_{Fib}(f)(0) &= 0 = O_{Fib}(g)(0) \\ O_{Fib}(f)(1) &= 1 = O_{Fib}(g)(1) \\ \forall n \in \mathbb{N} \setminus \{0,1\},\ O_{Fib}(f)(n) &= f(n-1) + f(n-2) \\ &\leq g(n-1) + g(n-2) \\ &= O_{Fib}(g)(n). \end{aligned}$$

Hence, $O_{Fib}(f) \leq_d O_{Fib}(g)$, as desired. □

By the definition of O_{Fib}, it is easy to see that the desired function Fib is the least fixpoint of O_{Fib}. The element $\mathrm{lfp}(O_{Fib})$ can also be constructed as the limit of the increasing sequence of functions $f_0, f_1, \ldots$ in $\boldsymbol{\mathcal{C}_{Fib}}$ obtained by iterating O_{Fib} on the bottom element of $\boldsymbol{\mathcal{C}_{Fib}}$. We write here the functions in the first iterations of the process:

$$f_0(n) := \bot \qquad f_1(n) := \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ \bot & \text{otherwise} \end{cases} \qquad f_2(n) := \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n \in \{1, 2\} \\ \bot & \text{otherwise} \end{cases}$$

The enrichment of the object space with $\bot$ allows us to deal with partially defined concepts, providing us with a suitable choice for the construction space. Nevertheless, such choice may be even more subtle, as we show next.

The Ackermann Function

Example 3.3.17 (Ackermann function)**.** *The Ackermann function was first introduced by Ackermann [3] as a function of three arguments, but the most commonly used version is the one with two arguments proposed by Peter [70] and later simplified by Robinson [74]. In the following, we will only consider the two-argument version, and we will refer to it as the Ackermann function.* $Ack\colon \mathbb{N}^2 \to \mathbb{N}$ *is defined recursively as follows:*

$$\left\lfloor \begin{array}{ll} \forall y \in \mathbb{N} : Ack(0, y) & := y + 1. \\ \forall x \in \mathbb{N} : Ack(x + 1, 0) & := Ack(x, 1). \\ \forall x, y \in \mathbb{N} : Ack(x + 1, y + 1) & := Ack(x, Ack(x + 1, y)). \end{array} \right\rfloor$$

The Ackermann function is a special case of recursive function for which the defined function is nested. This plays an important role in the choice of the construction space. The object space is again the set of functions with the right signature, namely $\mathbb{N}^2 \to \mathbb{N}$. Analogously to Example 3.3.13, the function Ack is defined on every element of its domain only after infinitely many steps. In fact, at each intermediate step of the recursion, we only have a partial function defined on a subset of $\mathbb{N}^2$. Hence, it may seem natural to consider as construction space the functions from $\mathbb{N}^2$ to $\mathbb{N}_\bot$ in order to allow the partially constructed functions to be undefined for some domain elements during construction. However, this enlargement of the construction space is not sufficient: due to the third rule of the definition, during the construction, the Ackermann function might be invoked on an output of a partially constructed object, which can possibly be $\bot$. Thus, the fact that the defined function occurs nested in itself prompts us to add $\bot$ to the *domain* of the functions in the construction space, and hence consider $\mathbb{N}_\bot \times \mathbb{N}_\bot$ as the domain of such functions. Notice that when the domain of the defined function consists of tuples of natural numbers the extension we make is not $(\mathbb{N} \times \cdots \times \mathbb{N})_\bot$ but rather $\mathbb{N}_\bot \times \cdots \times \mathbb{N}_\bot$.

Just as before, we can order this expanded space ${\mathbb{N}_\bot}^{\mathbb{N}_\bot \times \mathbb{N}_\bot}$ by the pointwise extension (also denoted $\leq_d$) of the definedness order $\leq_d$ on $\mathbb{N}_\bot$. However, the operator induced

by the definition of Ack is not monotonic on the full set of functions. Fortunately, it was shown that this operator is monotonic on a sufficiently large subset defined next. We expand the definedness order $\leq_d$ to $\mathbb{N}_\perp \times \mathbb{N}_\perp$ as the product order of $\leq_d$ on $\mathbb{N}_\perp$, and consider the subset $\mathcal{C}_{Ack}$ of monotonic functions of $\mathbb{N}_\perp^{\mathbb{N}_\perp \times \mathbb{N}_\perp}$. It turns out that the operator of the Ackerman definition, and the limit operation for increasing sequences of monotonic functions both preserve monotonitiy of functions, making $\mathcal{C}_{Ack} \subseteq \mathbb{N}_\perp^{\mathbb{N}_\perp \times \mathbb{N}_\perp}$ a suitable space to perform the induction process. In other words, we choose the construction space to be the cpo $\boldsymbol{\mathcal{C}_{Ack}} := \langle \mathcal{C}_{Ack}, \leq_d \rangle$. The proof of the fact that the construction space $\boldsymbol{\mathcal{C}_{Ack}}$ is a cpo analogous to the proof of Proposition 3.3.15, given that $\langle \mathbb{N}_\perp \times \mathbb{N}_\perp, \leq_d \rangle$ is a cpo, which follows easily from Lemma 3.3.14.

Proposition 3.3.18. *$\langle \mathbb{N}_\perp \times \mathbb{N}_\perp, \leq_d \rangle$ is a cpo.*

Proof. Follows easily from Lemma 3.3.14. □

Proposition 3.3.19. *$\boldsymbol{\mathcal{C}_{Ack}}$ is a cpo.*

Proof. Analogous to the proof of Proposition 3.3.15. □

It is important to note that for the first time, the injection θ is nontrivial, since we have enlarged the domain of the considered functions to $\mathbb{N}_\perp \times \mathbb{N}_\perp$. In particular, $\theta \colon \mathbb{N}^{\mathbb{N}^2} \to 2^{\mathcal{C}_{Ack}}$ sends a function $f \colon \mathbb{N}^2 \to \mathbb{N}$ to the set of functions

$$\theta(f) := \{g : (\mathbb{N}_\perp)^2 \to \mathbb{N}_\perp \mid \forall (x,y) \in \mathbb{N}^2, f(x,y) = g(x,y)\}.$$

Hence, it is easy to see that the surjective partial function $\pi \colon \mathcal{C}_{Ack} \to \mathbb{N}^{\mathbb{N}^2}$ associated with θ is defined only on the set

$$P := \{g \colon (\mathbb{N}_\perp)^2 \to \mathbb{N}_\perp \mid \forall (x,y) \in \mathbb{N}^2, g(x,y) \in \mathbb{N}\},$$

and sends a function $g \in P$ to the restriction $g\big|_{\mathbb{N}^2} \in \mathbb{N}^{\mathbb{N}^2}$ in the object space.

By the above recursive definition of Ack, the choice for the operator $O_{Ack} \colon \boldsymbol{\mathcal{C}_{Ack}} \to \boldsymbol{\mathcal{C}_{Ack}}$ is clear: for all $f \in L$,

$$O_{Ack}(f) := \begin{cases} (0,y) & \mapsto y+1 \\ (x+1,0) & \mapsto f(x,1) \\ (x+1,y+1) & \mapsto f(x,f(x+1,y)) \end{cases}$$

where $+$ is extended to $\mathbb{N}_\perp \times \mathbb{N}_\perp$ as in Example 3.3.13. Note that for any $f \in \boldsymbol{\mathcal{C}_{Ack}}$, $O_{Ack}(f)$ is indeed an element of $\boldsymbol{\mathcal{C}_{Ack}}$, since the composition of monotonic functions is monotonic.

Proposition 3.3.20. *The operator O_{Ack} is monotonic.*

Proof. Let $f, g \in \mathcal{C}_{Ack}$ such that $f \leq_d g$. Then we have

$$\begin{aligned}
&\forall y \in \mathbb{N}_\perp,\ O_{Ack}(f)(0,y) = y+1 = O_{Ack}(g)(0,y),\\
&O_{Ack}(f)(\perp,0) = f(\perp,0) \leq_d g(\perp,0) = O_{Ack}(g)(\perp,0),\\
&\forall x \in \mathbb{N} \setminus \{\perp,0\},\ O_{Ack}(f)(x,0) = f(x-1,0)\\
&\qquad\qquad\qquad\qquad \leq_d g(x-1,0) = O_{Ack}(g)(x,0).
\end{aligned}$$

Moreover, for all $x, y \in \mathbb{N}_\perp$,

$$\begin{aligned}
O_{Ack}(f)(x+1,y+1) &= f(x, f(x+1,y))\\
&\leq_d f(x, g(x+1,y))\\
&\leq_d g(x, g(x+1,y))\\
&= O_{Ack}(g)(x+1,y+1),
\end{aligned}$$

where the first inequality holds by the monotonicity of f. Hence, $O_{Ack}(f) \leq_d O_{Ack}(g)$, as desired. □

The construction process starts from the bottom element $\perp_{Ack}$ of $\boldsymbol{\mathcal{C}_{Ack}}$, namely the function $\perp_{Ack} \colon \mathbb{N}_\perp \times \mathbb{N}_\perp \to \mathbb{N}_\perp$ sending every pair of elements to $\perp$. By iteratively applying the operator O_{Ack}, we obtain an increasing sequence of monotonic functions $f_0, f_1, \ldots$ in $\boldsymbol{\mathcal{C}_{Ack}}$, representing the partial functions of the intermediate steps of the recursion. At the first steps of the process we get the functions defined on $(x,y) \in \mathbb{N}_\perp \times \mathbb{N}_\perp$ as follows:

$$f_0(n) := \perp \qquad f_1(x,y) := \begin{cases} y+1 & \text{if } x = 0 \\ \perp & \text{otherwise} \end{cases} \qquad f_2(x,y) := \begin{cases} y+1 & \text{if } x = 0 \\ 2 & \text{if } (x,y) = (1,0) \\ \perp & \text{otherwise} \end{cases}$$

Only after transfinitely many steps, we reach the least fixpoint of O_{Ack}. Finally, it is not hard to see that applying the projection π on the least fixpoint yields the defined object, i.e., $\pi(\mathrm{lfp}(O_{Ack})) = \mathrm{lfp}(O_{Ack})\big|_{\mathbb{N}^2} = Ack$.

We now move to examples of co-recursive definitions of functions. Once again, the choice of a suitable construction space turns out to be non-trivial.

Co-Recursive Function Definitions

Example 3.3.21 (Co-Fibonacci)**.** *The co-Fibonacci function* $co_Fib\colon \mathbb{N}^2 \to List$*, which maps a pair* (x, y) *of natural numbers to the Fibonacci sequence starting with* x, y*, is defined co-recursively to send* (x, y) *to the list starting with* x *and followed by* $co_Fib(y, x + y)$*.*

We can present the corecursive definition of co_Fib *by*

$$\lceil\ \forall x, y \in \mathbb{N} : co_Fib(x, y) := [x \mid co_Fib(y, x + y)].\ \rceil$$

In particular, $co_Fib(0, 1)$ is the list corresponding to the Fibonacci sequence, defined recursively in Example 3.3.13. The object space is again clear from the signature of the function we want to define: it is the set of functions from $\mathbb{N}^2$ to $List$.

As in Example 3.3.13, we need to enlarge the codomain of the considered functions in order to represent intermediate steps of the process. Thus, we define a set $List^o$, containing lists of natural numbers and finite lists of natural numbers ending with o. A list of the form $[x_1, \ldots, x_n \mid o]$ represents a list of natural numbers with an overdefined tail o. Intuitively, it identifies the set $\{[x_1, \ldots, x_n \mid l] \colon l \in List\}$ of fully defined lists of natural numbers. In other words, the element $[x_1, \ldots, x_n \mid o]$ conveys that all lists of the form $[x_1, \ldots, x_n \mid l]$ with $l \in List$ are still considered possible completions of the current (partially constructed) list. Intuitively, overdefined differs from undefined. The overdefined value signifies that something is defined, but its exact value remains unspecified. In contrast, an undefined value indicates the absence of definedness altogether. Consequently, a list with an undefined tail is itself undefined, whereas a list with an overdefined tail represents a defined list whose continuation is left indeterminate. [6] Accordingly, on $List^o$, the definedness order $\leq_d$ is defined inductively as follows:

- for all $t \in List^o$: $t \leq_d o$
- for all $x \in \mathbb{N}, t_1, t_2 \in List^o$: $[x \mid t_1] \leq_d [x \mid t_2]$ if $t_1 \leq_d t_2$

In this order, o is indeed "more defined" than any list. The set $List^o$ with the order $\leq_d$ is not a cpo since it has no least element, however, with the inverted order $\geq_d$ it indeed is a cpo, with "least" element o.

Lemma 3.3.22. $\langle List^o, \geq_d \rangle$ *is a cpo.*

Proof. Let $S \subseteq List^o$ be a chain. We have to show that S has a least upper bound. If S has finite cardinality, the claim is trivial. Suppose S has an infinite number of

[6] Note that the earlier introduced notation $[x \mid y]$ is used now to denote a list of $List^o$ with head a finite list x of natural numbers, and tail a list y of $List^o$.

elements. By the definition of $\geq_d$ and since S is a chain, if S contains an infinite list l, then l is the least upper bound of S. Suppose otherwise, i.e., all elements in S are finite lists of natural numbers or finite lists of natural numbers ending with o. We denote the length of a list $l \in S$ by $\mathit{length}(l) \in \mathbb{N}$. Note that if for some $l^1, l^2 \in S$ where $l^1 \neq l^2$, $\mathit{length}(l_1) = \mathit{length}(l_2)$, then l_1 must end with o and l^2 with a natural number, or vice versa. Therefore, S is well-ordered. Hence, for every $n \in \mathbb{N}$ we can identify l^n the least element of $\{l \in S \mid \mathit{length}(l) \geq_{\mathbb{N}} n\}$.

Let $l \in S$ be a list and $n \leq \mathit{length}(l)$, we denote by $l[n]$ the n-th element of l. We define an infinite list $L := [L_j]_{j \in \mathbb{N}}$ where

$$\forall j \in \mathbb{N} : L_j := l^{j+1}[j].$$

Notice that, for all $l, l' \in S$, the first $\min(\mathit{length}(l), \mathit{length}(l')) - 1$ elements of l and l' coincide. Hence, it is not hard to see that for any $l \in S$ we have $l \geq_d L$ by construction.

Let U be an upper bound for S, i.e., U is an infinite sequence of natural numbers such that $l \geq_d U$ for all $l \in S$. By the definition of the order, for all $l \in S$, the first $\mathit{length}(l) - 1$ elements of l and U coincide. Hence, it is easy to see that $L = U$. In particular, L is the least upper bound of S. □

The order $\geq_d$ can be extended in the standard, point-wise way to $(\mathit{List}^o)^{\mathbb{N}^2}$. We define the construction space $\boldsymbol{\mathcal{C}_{co_Fib}} = \langle (\mathit{List}^o)^{\mathbb{N}^2}, \geq_d \rangle$. The inversion of the definedness order, often used for recursion, mimics the order inversion between inductive and co-inductive definitions (hence the term *co-recursion*).

Proposition 3.3.23. *$\boldsymbol{\mathcal{C}_{co_Fib}}$ is a cpo.*

Proof. Let $S \subseteq \boldsymbol{\mathcal{C}_{co_Fib}}$ be a chain. By Lemma 3.3.22, we can define the function $f_{\vee S} \colon \mathbb{N}^2 \to \mathit{List}^o$ point-wise as follows: for each $x \in \mathbb{N}^2$, $f_{\vee S}(x) := \mathrm{lub}_{\mathit{List}}(\{f(x) \mid f \in S\})$. Since $\leq_{\mathit{List}^o}$ is defined point-wise, it is easy to see that $\mathrm{lub}(S) = f_{\vee S}$. □

Finally, we define the operator $O_{co_Fib} \colon \boldsymbol{\mathcal{C}_{co_Fib}} \to \boldsymbol{\mathcal{C}_{co_Fib}}$ by sending a function $f \in \boldsymbol{\mathcal{C}_{co_Fib}}$ to

$$O_{co_Fib}(f) \colon \mathbb{N}^2 \to \mathit{List}^o : (x, y) \mapsto [x \mid f(y, x + y)].$$

By the definition of $\geq_d$, it is easy to see that O_{co_Fib} is a monotonic operator. Moreover, the desired function co_Fib is the least fixpoint of the operator O_{co_Fib}. This coincides with the limit of the increasing sequence $f_0, f_1, \ldots$ constructed by iterating O_{co_Fib} on the bottom element $\bot_{\boldsymbol{\mathcal{C}_{co_Fib}}}$ of $\boldsymbol{\mathcal{C}_{co_Fib}}$, i.e., the function $\bot_{\boldsymbol{\mathcal{C}_{co_Fib}}} \colon \mathbb{N}^2 \to \mathit{List}^o$ sending every tuple to o. We report here the images of the functions in the first iterations of the process, depending on $(x, y) \in \mathbb{N}^2$:

$$f_0(x, y) = \bot_{\boldsymbol{\mathcal{C}_{co_Fib}}}(x, y) = o \qquad\qquad f_2(x, y) = [x, y \mid o]$$

$$f_1(x,y) = [x \mid o] \qquad\qquad f_3(x,y) = [x, y, x+y \mid o]$$

The limit of the above sequence is the co_Fib function, mapping pairs x, y to the infinite Fibonacci sequence starting at x and y. In general, from a recursive definition of a function it is possible to obtain a co-recursive definition, analogous as for co-Fibonacci. We make this precise in the following example.

Example 3.3.24. *If $\varphi : \mathbb{N} \to \mathbb{N}$ is a recursively defined function with m base cases and a recursive case defining $\varphi(n+m)$ in terms of the values $\varphi(n)$ up to $\varphi(n+m-1)$, we can generalize it to a class of functions φ_A where $A = (a_0, \ldots, a_{m-1}) \in \mathbb{N}^m$ is a tuple of values to be taken for the base cases, resulting in the recursive definition*

$$\left\lfloor \begin{array}{l} \varphi_A(0) := a_0. \\ \varphi_A(1) := a_1. \\ \vdots \\ \varphi_A(m-1) := a_{m-1}. \\ \forall n \in \mathbb{N} : \varphi_A(n+m) := g(\varphi_A(n), \ldots, \varphi_A(n+m-1)). \end{array} \right\rfloor$$

where g is some function from $\mathbb{N}^m$ to $\mathbb{N}$. In this case, it is possible to co-recursively define a function

$$co_\varphi \colon \mathbb{N}^m \to List \colon A \mapsto [\varphi_A(n)]_{n \in \mathbb{N}}$$

which sends an m-tuple A of base-case values to the list $[\varphi_A(0), \varphi_A(1), \ldots]$ corresponding to the image of φ_A.

As done previously, we can first represent the co-recursive definition of the function co_φ using a set of rules, namely

$$\left\lceil \begin{array}{l} \forall x_0, \ldots, x_{m-1} \in \mathbb{N} : co_\varphi(x_0, \ldots, x_{m-1}) := \\ \qquad\qquad [x_0 \mid co_\varphi(x_1, \ldots, x_{m-1}, g(x_0, \ldots, x_{m-1}))]. \end{array} \right\rceil$$

Similarly to what was observed for Example 3.3.21, the object space is the set of functions from $\mathbb{N}^m$ to $List$, and the construction space $\boldsymbol{\mathcal{C}_{co_\varphi}}$ is the cpo of functions from $\mathbb{N}^m$ to $List^o$ together with the point-wise order extension of $\leq_{List^o}$, denoted again by $\leq_{List^o}$. We can define a monotonic operator $O_{co_\varphi} \colon \boldsymbol{\mathcal{C}_{co_\varphi}} \to \boldsymbol{\mathcal{C}_{co_\varphi}}$ which maps a function $f \in \boldsymbol{\mathcal{C}_{co_\varphi}}$ to the function $O_{co_\varphi}(f) \colon \mathbb{N}^m \to List^o$, defined as

$$O_{co_\varphi}(f)(x_0, \ldots, x_{m-1}) := \\ [x_0 \mid f(x_1, \ldots, x_{m-1}, g(x_0, \ldots, x_{m-1}))],$$

For every tuple $(x_0, \ldots, x_{m-1}) \in \mathbb{N}^m$.

3.3.3 Definitions with Custom-Designed Cpo's

In this third and last subsection, we present a final example of a constructive definition of a function. Even though this definition deviates from the standard (co-)recursive account[7], it can indeed be formalized using our proposed framework.

Just like Example 3.3.11, the definition illustrated here falls under the company controls domain. Whereas Example 3.3.11 defined whether a company controls another, in this case, we want to define the *percentage of shares* of a company that another company controls. This is no longer a binary problem; instead the co-domain consists of an infinite set of values.

Example 3.3.25 (Controlled shares)**.** *If x and y are two companies, we say that x controls n shares of y if n is the sum of the shares of y owned by x or any company z of which x controls more than half of the shares.*

Let C be the set of companies. We can formally define the desired function $Csh\colon C^2 \to [0,1]$ as follows:

$$\left\{ \forall x \forall y : Csh(x,y) := \sum_{z \in \{x\} \cup \{u \mid Csh(x,u) > 0.5\}} Sh(z,y). \right\}$$

where $Sh\colon C^2 \to [0,1]$ is still the function mapping a pair of companies (x,y) to the fraction of shares of y owned by x.

The object space is the set of functions from S^2 to the interval $[0,1]$. The construction process is more complex than that of (co-)recursive definitions of functions. In fact, we now need to be able to decide whether $Csh(x,y) > 0,5$ is true before $Csh(x,y)$ is determined.

We can think about this construction process as a gradual refinement of each tuple's image. At the beginning of the process, we have no information about the image of Csh except that $Csh(x,y) \in [0,1]$ for all $x, y \in C$. At every rule application we get new information on the lower bounds of the images of elements of C^2. Since the upper bounds remain constant equal to 1, we may as well identify the interval in which an image is contained with its lower bound. By using this interpretation of the process, the choice for a construction space $\boldsymbol{\mathcal{C}_{Csh}}$ becomes clear, namely we consider the cpo of

[7] Interestingly, this example could be rephrased as a non-monotonic recursive definition. However, the treatment here is more natural and elegant.

functions from C^2 to $[0,1]$, with the pointwise extension of the standard order $\leq$ on real numbers.

Proposition 3.3.26. *$\mathcal{C}_{Csh}$ is a cpo.*

Proof. Since $([0,1], \leq)$ is a cpo, the proof is analogous to the proof of Proposition 3.3.15. □

On the cpo $\mathcal{C}_{Csh}$, we can consider the operator $O_{Csh} \colon \mathcal{C}_{Csh} \to \mathcal{C}_{Csh}$, which maps a function $f \colon C^2 \to [0,1]$ to $O_{Csh}(f)$, defined for every $(x,y) \in C^2$ by

$$O_{Csh}(f)(x,y) := \sum_{z \in \{x\} \cup \{u \mid f(x,u) > 0,5\}} Sh(x,y).$$

Proposition 3.3.27. *The operator O_{Csh} is monotonic.*

Proof. Let $f, g \colon C^2 \to [0,1]$ such that $f \leq_L g$. In particular, for all $x \in C$, we have $\{u \mid f(x,u) > 0,5\} \subseteq \{u \mid g(x,u) > 0,5\}$. Since $Sh(z,y) \geq 0$ for all $z, y \in C$, we have $O_{Csh}(f) \leq_L O_{Csh}(g)$, as desired. □

As anticipated, we can start the recursion from the bottom element of $\mathcal{C}_{Csh}$, namely the function f_0 sending every pair of companies to 0. By iteratively applying the operator O_{Csh} we get an increasing sequence of functions $f_0 \leq_{\mathcal{C}_{Csh}} f_1 \leq_{\mathcal{C}_{Csh}} f_2 \leq_{\mathcal{C}_{Csh}} \cdots$, whose limit is the desired defined function Csh and coincides with the least fixpoint of O_{Csh}. Notice that at any step t of the construction process, for each pair $(x,y) \in C^2$, the image $f_t(x,y)$ may not be the correct value of $Csh(x,y)$. Only in the last step, when the fixpoint is reached, certainty is reached of the correct value $Csh(x,y)$, for all pairs (x,y) at once. This is much unlike previous examples. This type of construction, using increasingly more precise bounds, lies at the basis of bound-founded ASP [6, 19].

3.4 More Advanced Constructive Definitions

Throughout this chapter, we focused on monotonic definitions involving a single concept. In contrast, real-world scenarios are often more complex. They may define multiple concepts in terms of one another, or they may result in non-monotonic operators, and sometimes it will be a combination of the two. Here we consider some more advanced definitions. Specifically, we focus on two categories: simultaneous constructions and nested constructions. Both deal with definitions in which multiple concepts are introduced at once. The difference lies in their structure: in simultaneous constructions the concepts are defined on equal footing, whereas in nested constructions they are organized hierarchically, with one definition acting as an auxiliary concept inside another.

3.4.1 Simultaneous Construction

We begin by considering definitions that define multiple concepts through simultaneous construction. As an illustration, we present an example of simultaneous induction, but the same principle holds for other (combinations of) types of constructive definitions. Simultaneous induction is so common and such a straightforward extension of induction that in general not a lot of words are spent on this generalization. It has been argued by Denecker et al. [33] that the informal semantics of a logic program is an inductive definition. Many logic programs include several predicate symbols in their rule heads. Such programs can be understood as simultaneous inductive definitions, where multiple predicates are defined together in a mutually dependent way.

Example 3.4.1. *The set of even natural numbers $Even$ and the set of odd natural numbers Odd are defined by simultaneous induction:*

- $0 \in Even$.
- *If* $x \in Even$, *then* $x + 1 \in Odd$.
- *If* $x \in Odd$, *then* $x + 1 \in Even$.

This definition is formally represented by the following set of rules:

$$\left[\begin{array}{l} \forall x : Even(x) \leftarrow x = 0. \\ \forall x : Even(x+1) \leftarrow Odd(x). \\ \forall x : Odd(x+1) \leftarrow Even(x). \end{array}\right]$$

We expect that both $Even$ and Odd are sets of natural numbers, thus, for each the object space is equal to the set of all sets over natural numbers, i.e., $2^{\mathbb{N}}$. We can straightforwardly combine the two spaces by taking the Cartesian product $2^{\mathbb{N}} \times 2^{\mathbb{N}}$ as the object space $\boldsymbol{\mathcal{O}_{EO}}$. For the construction space we simply equip the object space with the product order representing simultaneous induction, i.e., $\boldsymbol{\mathcal{C}_{EO}} = \langle 2^{\mathbb{N}} \times 2^{\mathbb{N}}, \subseteq^2 \rangle$, where $(A_1, B_1) \subseteq^2 (A_2, B_2)$ iff $A_1 \subseteq A_2$ and $B_1 \subseteq B_2$. It is well-known that the product-lattice of two complete lattices is again a complete lattice. Just like in all previous examples, the operator $O_{EO} : \boldsymbol{\mathcal{C}_{EO}} \to \boldsymbol{\mathcal{C}_{EO}}$ corresponds to rule application, mapping a pair of sets $(S_E, S_O) \in \boldsymbol{\mathcal{C}_{EO}}$ to

$$O_{EO}(S_E, S_O) := (O_{Even}(S_E, S_O), O_{Odd}(S_E, S_O)),$$

where O_{Even} and O_{Odd} are the operators corresponding to rule application of all rules with respectively $Even$ and Odd in the head:

$$O_{Even}(S_E, S_O) := \{0\} \cup \{x + 1 \mid x \in S_O\},$$

$$O_{Odd}(S_E, S_O) := \{x + 1 \mid x \in S_E\}.$$

Proposition 3.4.2. *The operator O_{EO} is monotonic.*

Proof. Let $(S_E, S_O) \subseteq^2 (S'_E, S'_O)$, then $S_E \subseteq S'_E$ and $S_O \subseteq S'_O$. Therefore, for every $x \in O_{Even}(S_E, S_O)$ either $x = 0$ or $x - 1 \in S_O$. If $x = 0$, then $x \in O_{Even}(S'_E, S'_O)$. Else, as $S_O \subseteq S'_O$, $x - 1 \in S'_O$, hence $x \in O_{Even}(S'_E, S'_O)$. Thus $O_{Even}(S_E, S_O) \subseteq O_{Even}(S'_E, S'_O)$. Analogously, we find $O_{Odd}(S_E, S_O) \subseteq O_{Odd}(S'_E, S'_O)$. In conclusion, $O_{EO}(S_E, S_O) \subseteq^2 O_{EO}(S'_E, S'_O)$. □

From its definition it is easy to derive that in the i-th iteration of O_{EO} the natural number $i - 1$ is added to the appropriate set. Thus, the fastest monotonic induction is represented by the sequence

$$(\emptyset, \emptyset) \subseteq^2 (\{0\}, \emptyset) \subseteq^2 (\{0\}, \{1\}) \subseteq^2 (\{0, 2\}, \{1\}) \subseteq^2 \ldots \longrightarrow (Even, Odd)$$

Clearly, the same principles apply for simultaneous definitions as for singular constructive definitions. The object space, construction space and semantic operator appear naturally from the Cartesian product.

3.4.2 Nested Construction

Just like simultaneous construction, nested constructive definitions [79, 65] involve reasoning about multiple concepts at once. However, rather than treating all concepts equally, nested definitions introduce a hierarchy, i.e., some definitions serve as auxiliary constructs embedded within others. Interestingly, a nested definition may combine different types of constructive definitions. For example, an inductive definition can be nested inside a co-inductive one, or vice versa.

As the hierarchy between definitions must be respected during the construction process, this requires a more elaborate formalization. Specifically, in a nested definition, the innermost or auxiliary concept functions somewhat like a parameter concept: it is evaluated relative to the outermost definition. However, unlike a fixed parameter, its value is not independent. Instead, it corresponds to the defined object of another definition that depends on the current stage of the main concept. In other words, as the outer definition evolves, the auxiliary concept must be recomputed at each step, using the current value for the main concept as context. The next example illustrates an inductive definition nested inside a co-inductive one, though other combinations or deeper nestings of constructive definitions can also occur.

Example 3.4.3. *A list l is fair with respect to a fixed natural number n if n occurs infinitely often in l.*

This is a fairly simply natural language statement. However, to formalize this definition as a set of constructive rules, we require induction over the list-structure to determine the number of occurrences of n in a list and co-induction over the number of occurrences of n following the requirement that this number is infinite. This results in the nested formal definition:

$$\left[\begin{array}{l} Fair^c(l) \leftarrow Fair^i(l) \\ \left|\begin{array}{l} Fair^i([n \mid t]) \leftarrow Fair^c(t) \\ Fair^i([h \mid t]) \leftarrow Fair^i(t) \end{array}\right. \end{array}\right]$$

Assume a fixed value for $Fair^c$. Naturally, the inductive definition of $Fair^i$ then derives each list of the form

$$l_1 +\!\!+ [n \mid Nil] +\!\!+ l_2$$

with $+\!\!+$ the concatenation operator gluing lists together, and with l_1 an arbitrary finite list and l_2 in $Fair^c$. Consequently, we see that for every list l^i in $Fair^i$ there must exist a list l^c in $Fair^c$ such that l^i contains strictly more occurrences of n than l^c. Then the constructed value for $Fair^i$ is used to update the value of $Fair^c$.

With this observation it is easy to see how the construction process for the main co-inductive definition will evolve. It starts from the entire domain, after one iteration, we lose all lists with no occurrences of n. After the next iteration, $Fair^c$ does not contain any lists with one or less occurrences of n, and so on.

At the fixpoint, after infinitely many steps, we have successfully eliminated all lists with a finite number of occurrences of n. In contrast, all lists in which n occurs infinitely many times are never deleted from our set.

More formally, the innermost definition can be treated as a normal definition with the outermost defined concept as a fixed parameter in the definition. Hence, their object spaces and construction spaces are derived as usual. Here, the object space corresponds to the set of lists of natural numbers 2^{List}, while the construction space $\boldsymbol{\mathcal{C}_{Fair^i}}$ is the object space endowed with the subset order, as the innermost definition is an inductive definition. Parallel to the previous example, the semantic operator $O^F_{Fair^i}$ for $Fair^i$ is the result of rule application of all rules in the definition of $Fair^i$ where $Fair^c$ is treated as a parameter-concept with a fixed value F. Now we want to assign the defined object of the innermost definition as the value for the nested parameter concept. This results in the following semantic operator $O_{Fair^c} : \boldsymbol{\mathcal{C}_{Fair^c}} \rightarrow \boldsymbol{\mathcal{C}_{Fair^c}}$, which maps a set of lists of natural numbers S to:

$$O_{Fair^c}(S) := \mathrm{lfp}(O^S_{Fair^i})$$

Proposition 3.4.4. *The operator O_{Fair^c} is monotonic.*

Proof. Let $S_1 \supseteq S_2$. Let $S_3 = \mathrm{lfp}(O^{S_2}_{Fair^i})$. We show that for every L_α in the fastest monotone induction $(L_i)_{i\geq 0}$ of $O^{S_1}_{Fair^i}$, $L_\alpha \supseteq S_3$. For limit ordinals, this is trivial. On the other hand, for the case of successor ordinals $i+1$ let $[h \mid t] \in S_3$. Then we know either (1) $h = n$ and $t \in S_2$, or (2) $t \in S_3$. For the first case, as $S_1 \supseteq S_2$, $t \in S_1$. Therefore, $[h \mid t] \in L_{i+1}$. For the second case it suffices to see that if $L_i \supseteq S_3$ (as by th induction hypothesis), then $t \in L_i$, hence $t \in L_{i+1}$. As a result, $L_{i+1} \supseteq S_3$. This holds for every L_α in the fastest monotone induction $(L_i)_{i\geq 0}$, therefore $\mathrm{lfp}(O^{S_1}_{Fair^i}) \supseteq \mathrm{lfp}(O^{S_2}_{Fair^i})$. □

Following the informal explanation, the co-induction process is given by the sequence:

$$List = Fair^0 \supseteq Fair^1 \supseteq \ldots \supseteq Fair^i \supseteq \ldots \rightarrow Fair$$

with $Fair^i$ the set of all lists l such that n occurs at least i times in l. Thus, at step i of the co-induction process all lists l with exactly $i-1$ occurrences of n are removed from the defined set. In the limit, only the lists in which n occurs infinitely often remain.

This example illustrates how nested definitions naturally arise when the construction of one concept depends on another definition that must itself be fully resolved at each step. Such cases are more intricate than simultaneous definitions, as they impose a hierarchical structure between definitions and require careful handling of the dependencies involved. Nevertheless, they highlight the expressive power of constructive definitions and their ability to capture the layered reasoning processes common in constructive knowledge representation. In the approach for nested definitions developed here, each step of the construction process fully evaluates the auxiliary (inner) concept before updating the outer concept. This means the inner definition is essentially treated like a parameter that is recalculated at every stage. However, there is an alternative viewpoint that brings this closer to the style of simultaneous definitions: we can think of the nested construction as driven by component operators that act together, but where the inner operator is subordinated to the outer one through a hierarchical or lexicographical ordering. However, such an operator is typically non-monotonic. This provides further motivation to explore AFT, which is capable of handling such non-monotonic operators. At the same time, we believe our view better reflects the underlying intuition behind the construction process, as it captures more accurately how the auxiliary concept is recomputed and integrated at each stage.

3.5 Conclusion and Future Work

We investigated a heterogeneous set of monotonic constructive definitions, coming from different domains and never brought together before, in a uniform framework. Our analysis confirms the power of fixpoint theory for abstract formalization, but also points to a key distinguishing factor: the construction space, the set of objects that serve as approximations of the object being defined. We propose the general term *monotonic constructive definitions* for a class of definitions that includes recursive and inductive definitions and developed a framework that clearly emphasizes how different types of definitions can be classified according to different types of construction spaces.

This is a crucial step towards the development of knowledge representation languages that include a variety of constructive definitions. Our framework suggests such language requires a formal syntax for definitions such that one can automatically and uniformly derive a suitable object space, semantic operator and construction space. As shown by the examples, while the first two are straightforward, the latter may be non-trivial. We have illustrated different types of definitions by example, allowing us to handpick the most convenient, natural construction space. The challenge is that a uniformly derived construction space needs to be strong enough to handle all considered definitions, and the defined object should coincide with the one obtained with the handpicked construction space. This chapter offers an important first step towards solving this issue by classifying different types of definitions based on the kind of construction space they require. This means identifying the correct type of definition will be an essential part of the syntax of the considered knowledge representation language.

By no means do we claim our list of types of constructive definitions to be exhaustive. Another type of constructive definitions, not considered here, are non-monotonic "iterated" inductive definitions which have been researched in mathematical logic [42, 61, 17]. In iterated inductive definitions, e.g., inductive definitions over a well-founded order, multiple objects are inductively defined in terms of other defined objects on a lower or equal level. Whereas nested definitions make the hierarchy explicit, in general, for iterated inductive the ordering might be implicit. Once all objects on some level are well-defined, their values can be used to derive the value of any object on a higher level. This is the natural principle of *stratification*. It has been argued that this principle is implemented by the well-founded semantics of logic programming [33, 29, 31]. Thus, the declarative logic underlying logic programming can be seen as a logic of this type of constructive definition.

In this chapter, we focused on *monotonic* constructive definitions. Non-monotonic *inductive* definitions have been studied intensively, including in a fixpoint-theoretic setting (known as *Approximation Fixpoint Theory (AFT)* [32]). In the terminology of the current chapter, dealing with this non-monotonicity requires switching to a different construction space (a space of approximations). A natural next question we

wish to tackle is whether this framework can also be of use for studying non-monotonic *recursive* definitions.

As a final remark, we argued that constructive definitions are an important form of human knowledge. Of course, many other types of knowledge are important as well. Contrary to the languages of logic and functional programming, which support mainly definitions used as programs, expressive KR languages should offer language constructs for expressing a broad range of knowledge. One such example is the logic FO($\cdot$), which extends FO with among others an expressive rule-based language construct for definitional knowledge, inspired by logic programming [26, 25].

Chapter 4

Analyzing Semantics of Aggregate Answer Set Programming

4.1 Introduction

In the previous chapter, we claimed that constructive definitions form a crucial type of knowledge. What we did not fully address, however, is how to formally describe such definitions within a logical system providing a clear syntax which systematically yields the object space, the construction space, and the semantic operator. For some classes of constructive definitions, such logics already exist. For instance, Denecker et al. [33] argues that a normal logic program can be viewed as an inductive definition, framing normal logic programming (LP) as a logic of inductive definitions. As LP is build upon a fairly simple base logic, the range of inductive definitions that can be expressed through normal logic programs is limited.

We introduce a framework for defining semantics for extensions of LP, and in particular answer set programming (ASP), using Approximation Fixpoint Theory (AFT). Our focus will be on aggregates, a construct that continues to inspire significant discussion in the ASP community, as many approaches exist, but so far no consensus on how to handle aggregates in those logics has been reached.

Real-world applications often demand aggregate-constructs to reason about sets of data, counts, sums, or other collective properties. Aggregate expressions are very useful and have been added to classical logic, query languages, constraint languages, and

also to LP and ASP. The effort it takes to add aggregates to (syntax and semantics of) a logic is very language dependent. E.g., to extend first order logic (FO) with a *Count* aggregate, we extend the definition of "term" with a new inductive rule: "*If* $a_1, \ldots, a_n$ *are variables, and* ψ *a formula, then* $Count(\{(a_1, \ldots, a_n), \psi\})$ *is a term*" and the definition of "interpretation of a term t in structure $\mathcal{I}$" (used in the definition of $\models$) with: "*If* $t = Count(\{(a_1, .., a_n), \psi\})$ *then* $t^{\mathcal{I}}$ *is* $\#\{(d_1, \ldots, d_n) \in Dom(\mathcal{I})^n \mid \mathcal{I}[a_1 : d_1, \ldots, a_n : d_n] \models \psi\}$*, i.e., the number of tuples that satisfy* ψ *in* $\mathcal{I}$.". The method is simple and follows Frege's compositionality principle.

In LP and ASP, it is much more difficult. Research into extensions with aggregates (well-founded and stable semantics) started with the work of Kemp and Stuckey [53]. Since then, many (disagreeing) proposals have spawned from the field. One approach applies AFT to aggregate LP and ASP, resulting in the ultimate stable and well-founded semantics in the work by Denecker et al. [34], and in the broader framework by Pelov et al. [69] where stable and well-founded semantics are induced by a choice of a three-valued truth function. Interestingly, by offering a range of approximators, Pelov's work seemingly covers multiple proposals.

Here, we clarify and expand this work. First, we show how each three-valued truth assignment can be broken up in a lower and an upper ternary satisfaction relation which, in the context of ASP, can be easily related to the reduct approach originally used by Gelfond and Lifschitz [46] to define stable semantics. Then we focus on aggregate ASP using examples from the literature. Where possible, we consider aggregate atoms with positive and negative literals as conditions. However, the semantics described by Gelfond and Zhang [47] does not allow *negation by default* inside an aggregate atom, therefore we only consider positive conditions for this specific case. It is shown that not only the semantics of Denecker et al. [34], Pelov et al. [69] but also those of Liu et al. [57] and Gelfond and Zhang [47] are instances of the framework. But not all proposed semantics for aggregate ASP semantics belong to the framework; for example those of Ferraris [43], Marek and Remmel [58], Faber et al. [41]. We investigate the reason for this. This work contributes to the discussion about semantics for extensions of ASP by clarifying some important principles of non-monotonic reasoning and constructive knowledge, and by showing where they are applied and where other principles are applied.

This chapter is mainly based on Vanbesien et al. [85], but adapted to fit in the current formalization.

4.2 Logic Programming and Approximation Fixpoint Theory

We start by sketching how to use AFT to define constructive semantics for programs based on some logic $\mathscr{L}$.[1] We assume $\mathscr{L}$ is a first-order logic with a (Herbrand) model semantics defined using a two-valued truth function $\mathcal{H}^2$, or equivalently, a satisfaction relation $\models_2$, where $\mathcal{I} \models_2 \phi$ iff $\mathcal{H}^2_{\mathcal{I}}(\phi) = \mathbf{t}$. An $\mathscr{L}$-program is then defined as follows:

Definition 4.2.1 ($\mathscr{L}$-program)**.** *An $\mathscr{L}$-program over a set of propositional atoms Σ is a set of rules r of the form $p \leftarrow \psi$ such that the body ψ is a formula of $\mathscr{L}$ and the head $p \in \Sigma$ is a propositional atom. Given a rule r we use the terms $head(r)$ and $body(r)$ to respectively denote the head and the body of the rule.*

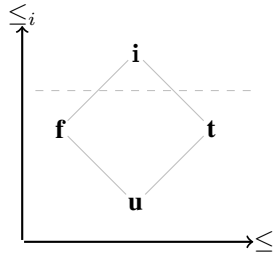
Importantly, the rule operator '$\leftarrow$' is not a connective of $\mathscr{L}$; instead AFT defines its meaning as a *construction operator*. LP and (non-disjunctive) ASP are instances of this where $\mathscr{L}$ is simply the logic of conjunctions of literals p or $\neg p$ under standard interpretation.

Definition 4.2.2. *Given an $\mathscr{L}$-program P, the corresponding exact space $\boldsymbol{\mathcal{E}_P} = \langle \mathcal{E}_P, \leq \rangle$ is the set of P's Herbrand interpretations ordered by the point-wise extension of the truth order ($\mathbf{f} < \mathbf{t}$).*

We assume that a two-valued Herbrand interpretation $\mathcal{I}$ over Σ is represented by the subset of Σ that evaluates to $\mathbf{t}$ in $\mathcal{I}$. The point-wise extension of the truth-order then corresponds to the subset-relation. More precisely, $\boldsymbol{\mathcal{E}_P} = \langle 2^{\Sigma}, \subseteq \rangle$. Consequently, it is not hard to see that $\boldsymbol{\mathcal{E}_P}$ is a complete lattice.

The iAFT and sAFT approximation spaces $\boldsymbol{\mathcal{E}_P}^c$ and $\boldsymbol{\mathcal{E}_P}^2$ then correspond to three- and four-valued interpretations. Specifically, any three- or four-valued interpretation $\mathfrak{A}$ can be split into a pair $(\mathcal{I}, \mathcal{J})$ by splitting truth values as in $\mathbf{t} \sim (\mathbf{t}, \mathbf{t}), \mathbf{f} \sim (\mathbf{f}, \mathbf{f}), \mathbf{u} \sim (\mathbf{f}, \mathbf{t})$ and $\mathbf{i} \sim (\mathbf{t}, \mathbf{f})$. For example, if some propositional atom $p \in \Sigma$ evaluates to $\mathbf{u}$ in $\mathfrak{A}$, then $\mathcal{H}^2_{\mathcal{I}}(p) = \mathbf{f}$ and $\mathcal{H}^2_{\mathcal{J}}(p) = \mathbf{t}$, and vice versa. If $\mathfrak{A}$ is three-valued, then $\mathcal{I} \subseteq \mathcal{J}$, and $\mathcal{I}$ is a lower bound, while $\mathcal{J}$ is an upper bound of $\mathfrak{A}$. The many-valued truth-value sets are typically ordered by a truth-order $\leq$, extending the truth-order on the bivalent truth-value set, and by an information-order $\leq_i$ as visualized by the following Hasse diagram.

[1] For ease of discussion, this work only considers Herbrand interpretations and ground programs.



Then the three- and four-valued structures are equipped with the point-wise extensions of said truth order $\leq$ and information order $\leq_i$, which, respectively, isomorphically correspond to the truth order $\leq_t$ and $\leq_p$ on $\boldsymbol{\mathcal{E}_P}^2$ and $\boldsymbol{\mathcal{E}_P}^c$. As a result, any $\leq$- or $\leq_i$-monotonic operator Γ on the two-, three- or four-valued structures respectively corresponds to a $\leq_t$- or $\leq_p$-monotonic operator on $\boldsymbol{\mathcal{E}_P}, \boldsymbol{\mathcal{E}_P}^c, \boldsymbol{\mathcal{E}_P}^2$.

An $\mathcal{L}$-program P is characterized by the immediate consequence operator $T_P : \boldsymbol{\mathcal{E}_P} \to \boldsymbol{\mathcal{E}_P}$. This operator fulfills the role of O, the approximated operator.

Definition 4.2.3. *The immediate consequence operator $T_P : \boldsymbol{\mathcal{E}_P} \to \boldsymbol{\mathcal{E}_P}$ for a program P is such that $T_P(\mathcal{I}) = \mathcal{J}$ if for every ground atom p,*

$$\mathcal{H}^2_{\mathcal{J}}(p) = \mathrm{lub}_{\leq}(\{\mathcal{H}^2_{\mathcal{I}}(\psi) \mid (p \leftarrow \psi) \in P\}).$$

There are two ways to define AFT semantics using an approximator A_P for T_P. One way is to use $A_P = U_{T_P}$, the most precise approximator of T_P in $\boldsymbol{\mathcal{E}_P}^c$ leading to ultimate versions of the family of AFT semantics as described by Denecker et al. [36] and used by Denecker et al. [34] for defining semantics of Aggregate LP. This is the most precise approach, but computationally costly.

The other way is to extend $\mathcal{L}$'s truth assignment to three- or four-valued interpretations and by interpreting T_P's definition in this broader context. E.g., a three-valued truth assignment $\mathcal{H}^3$ induces a three-valued immediate consequence operator Φ_P where $\Phi_P(\mathfrak{A}) = \mathfrak{A}'$ if for every atom p,

$$\mathcal{H}^3_{\mathfrak{A}'}(p) = \mathrm{lub}_{\leq}\{\mathcal{H}^3_{\mathfrak{A}}(\psi) \mid (p \leftarrow \psi) \in P\}.$$

Such operator Φ_P corresponds isomorphically to an $\boldsymbol{\mathcal{E}_P}^c$-operator A_P on pairs $\mathcal{I} < \mathcal{J}$ of two-valued interpretations. But for A_P to be an approximator of T_P, the three-valued truth assignment $\mathcal{H}^3$ should satisfy a condition introduced for three-valued logic by Kleene [55]:

Definition 4.2.4 (Regular truth assignment)**.** *A three-valued truth assignment $\mathcal{H}^3$ of $\mathcal{L}$ is* regular *iff for all formulas ψ, for all three-valued structures $\mathfrak{A}$ interpreting ψ:*

- *if $\mathfrak{A}$ is two-valued, then $\mathcal{H}^2_{\mathfrak{A}}(\psi) = \mathcal{H}^3_{\mathfrak{A}}(\psi)$ (i.e., $\mathcal{H}^3$ extends $\mathcal{H}^2$), and*

- *if* $\mathfrak{A} \leq_p \mathfrak{A}'$ *then* $\mathcal{H}^3_{\mathfrak{A}}(\psi) \leq_p \mathcal{H}^3_{\mathfrak{A}'}(\psi)$ *(i.e.,* $\mathcal{H}^3$ *is* $\leq_p$*-monotonic)*

For the four-valued case, we additionally require symmetry.

E.g., Kleene's strong three-valued truth assignment $\mathcal{H}^{SK}$ of FO (introduced by Kleene [55]) and Belnap's four-valued extension (introduced by Belnap [7]) are regular. They induce multi-valued extensions Φ_P of T_P first introduced by Fitting [45]. Later, Φ_P was found to correspond to an approximator A_P of T_P whose KK, WF and stable fixpoints correspond to the semantics of the same name.

In LP and ASP, it is often taken for granted that LP's non-monotonicity is due to its non-classical negation not. But it is evident in AFT-based semantics, that the main non-classical connective is the rule operator: its semantics is defined via operators and constructive processes while negation in bodies is treated like the other FO connectives, using three-valued logic only for approximation of the standard classical connectives.

The AFT road is quite unlike other semantic techniques in ASP. In the next section, we reformulate the framework in more accessible terms for the ASP community.

4.3 Ternary Satisfaction Relations

Ternary satisfaction relations were used originally by Liu et al. [57] in the context of Aggregate ASP semantics where they were called *sub-satisfiability relations*.

We still assume a base logic $\mathscr{L}$ equipped with satisfaction relation $\models_2$. Below, we restrict ourselves to three-valued interpretations corresponding to pairs $(\mathcal{I}, \mathcal{J})$ of two-valued Herbrand interpretations $\mathcal{I} \subseteq \mathcal{J}$ (but extension to non-Herbrand interpretations is possible).

Definition 4.3.1 (Ternary satisfaction relation)**.** *A* ternary satisfaction relation *(TSR)* $\models_3$ *of* $\mathscr{L}$ *is a relation between pairs* $(\mathcal{I}, \mathcal{J})$ *of interpretations such that* $\mathcal{I} \subseteq \mathcal{J}$*, and formulas* ψ *of* $\mathscr{L}$ *such that*

$$\mathcal{I} \models_2 \psi \textit{ iff } (\mathcal{I}, \mathcal{I}) \models_3 \psi.$$

Definition 4.3.2. *A* ternary satisfaction relation *(TSR)* $\models_3$ *of* $\mathscr{L}$ *is* lower-monotonic *if for all formula* ψ *of* $\mathscr{L}$ *and for all interpretations* $\mathcal{I}$*,* $\mathcal{J}$*,* $\mathcal{I}'$ *such that* $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$*:*

$$(\mathcal{I}, \mathcal{J}) \models_3 \psi \textit{ implies } (\mathcal{I}', \mathcal{J}) \models_3 \psi.$$

It is lower-regular *if for all formula* ψ *of* $\mathscr{L}$ *and for all interpretations* $\mathcal{I}$*,* $\mathcal{J}$*,* $\mathcal{I}'$*,* $\mathcal{J}'$ *such that* $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}' \subseteq \mathcal{J}$ *(*$(\mathcal{I}, \mathcal{J}) \leq_p (\mathcal{I}', \mathcal{J}')$*):*

$$(\mathcal{I}, \mathcal{J}) \models_3 \psi \textit{ implies } (\mathcal{I}', \mathcal{J}') \models_3 \psi.$$

It is upper-regular *if for all formula ψ of $\mathfrak{L}$ and for all interpretations $\mathcal{I}$, $\mathcal{J}$, $\mathcal{I}'$, $\mathcal{J}'$ such that $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}' \subseteq \mathcal{J}$ ($(\mathcal{I},\mathcal{J}) \leq_p (\mathcal{I}',\mathcal{J}')$):*

$$(\mathcal{I}',\mathcal{J}') \models_3 \psi \textit{ implies } (\mathcal{I},\mathcal{J}) \models_3 \psi.$$

Proposition 4.3.3. *If $\models_3^{\boldsymbol{\mathcal{L}}}$ is a lower-, and $\models_3^{\boldsymbol{\mathcal{U}}}$ is an upper-regular TSR, then $\models_3^{\boldsymbol{\mathcal{L}}} \subseteq \models_3^{\boldsymbol{\mathcal{U}}}$.*

Proof.

$$\begin{aligned}(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi &\text{ implies } (\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{L}}} \psi \\ &\text{implies } \mathcal{I} \models_2 \psi \\ &\text{implies } (\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{U}}} \psi \\ &\text{implies } (\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{U}}} \psi\end{aligned}$$

□

Also, a three-valued truth-function $\mathcal{H}^3$ corresponds one to one to pairs $(\models_3^{\boldsymbol{\mathcal{L}}}, \models_3^{\boldsymbol{\mathcal{U}}})$ of *TSRs*. The correspondence is (1) $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ iff $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) = \mathbf{t}$, and (2) $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$ iff $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) \in \{\mathbf{t},\mathbf{u}\}$.

Proposition 4.3.4. *$\mathcal{H}^3$ is regular iff $\models_3^{\boldsymbol{\mathcal{L}}}$ is a lower- and $\models_3^{\boldsymbol{\mathcal{U}}}$ is an upper-regular TSR.*

Proof. Let $\mathcal{H}^2$ denote the two-valued truth assignment.

1. Assume $\mathcal{H}^3$ is regular; we show that $\models_3^{\boldsymbol{\mathcal{L}}}$ is a lower-regular *TSR*:
   - $\mathcal{H}^2_{\mathcal{I}}(\psi) = v$ with $v \in \{\mathbf{f},\mathbf{t}\}$, iff $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) = v$. This entails that $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) = \mathbf{t}$ iff $\mathcal{H}^2_{\mathcal{I}}(\psi) = \mathbf{t}$. So, $(\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ iff $\mathcal{I} \models_2 \psi$. Hence, $\models_3^{\boldsymbol{\mathcal{L}}}$ is a *TSR*.
   - If $(\mathcal{I},\mathcal{J}) \leq_p (\mathcal{I}',\mathcal{J}')$ then for every ψ : $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) \leq_p \mathcal{H}^3_{(\mathcal{I}',\mathcal{J}')}(\psi)$. So, if $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) = \mathbf{t} = (\mathbf{t},\mathbf{t})$, the truth-value according to $\mathcal{H}^3_{(\mathcal{I}',\mathcal{J}')}(\psi)$ should also be equal to $\mathbf{t}$. Consequently $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ implies $(\mathcal{I}',\mathcal{J}') \models_3^{\boldsymbol{\mathcal{L}}} \psi$. Therefore, $\models_3^{\boldsymbol{\mathcal{L}}}$ is lower-regular.
2. Assume $\mathcal{H}^3$ is regular; we show that $\models_3^{\boldsymbol{\mathcal{U}}}$ is an upper-regular *TSR*:
   - $\mathcal{H}^2_{\mathcal{I}}(\psi) = v$ with $v \in \{\mathbf{f},\mathbf{t}\}$, iff $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) = v$. This entails that $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) = \mathbf{f}$ iff $\mathcal{H}^2_{\mathcal{I}}(\psi) = \mathbf{f}$. In other words, $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) \neq \mathbf{f}$ iff $\mathcal{H}^2_{\mathcal{I}}(\psi) \neq \mathbf{f}$. Based on the derivation of $\models_3^{\boldsymbol{\mathcal{U}}}$ from the truth assignment,

it is clear that $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) \neq \mathbf{f}$ iff $(\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$. On the other hand, for the two-valued truth-assignment, it holds that $\mathcal{H}^2_{\mathcal{I}}(\psi) \neq \mathbf{f}$ iff $\mathcal{H}^2_{\mathcal{I}}(\psi) = \mathbf{t}$ iff $\mathcal{I} \models_2 \psi$. So, $(\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$ iff $\mathcal{I} \models_2 \psi$. Hence $\models_3^{\boldsymbol{\mathcal{U}}}$ is a *TSR*.

- If $(\mathcal{I},\mathcal{J}) \leq_p (\mathcal{I}',\mathcal{J}')$ then for every ψ : $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) \leq_p \mathcal{H}^3_{(\mathcal{I}',\mathcal{J}')}(\psi)$. So, if $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) = \mathbf{f} = (\mathbf{f},\mathbf{f})$, the truth-value according to $\mathcal{H}_{(\mathcal{I}',\mathcal{J}')^3}(\psi)$ should also be equal to $\mathbf{f}$. Consequently $(\mathcal{I},\mathcal{J}) \not\models_3^{\boldsymbol{\mathcal{U}}} \psi$ implies $(\mathcal{I}',\mathcal{J}') \not\models_3^{\boldsymbol{\mathcal{U}}} \psi$. Therefore, $(\mathcal{I}',\mathcal{J}') \models_3^{\boldsymbol{\mathcal{U}}} \psi$ implies $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$. Thus, $\models_3^{\boldsymbol{\mathcal{U}}}$ is upper-regular.

3. Assume $\models_3^{\boldsymbol{\mathcal{L}}}$ is a lower- and $\models_3^{\boldsymbol{\mathcal{U}}}$ is an upper-regular *TSR*; we show that $\mathcal{H}^3$ is regular.

   - $(\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ iff $\mathcal{I} \models_2 \psi$ iff $(\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$. Therefore $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) = \mathbf{t}$ iff $\mathcal{H}^2_{\mathcal{I}}(\psi) = \mathbf{t}$ iff $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) \in \{\mathbf{t},\mathbf{u}\}$. Consequently, $\mathcal{H}^3_{(\mathcal{I},\mathcal{I})}(\psi) = \mathbf{f}$ iff $\mathcal{H}^2_{\mathcal{I}}(\psi) = \mathbf{f}$. Hence $\mathcal{H}^3$ coincides with $\mathcal{H}^2$ for two-valued interpretations.
   - If $(\mathcal{I},\mathcal{J}) \leq_p (\mathcal{I}',\mathcal{J}')$, then $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ implies $(\mathcal{I}',\mathcal{J}') \models_3^{\boldsymbol{\mathcal{L}}} \psi$. Therefore $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) = \mathbf{t}$ implies $\mathcal{H}^3_{(\mathcal{I}',\mathcal{J}')}(\psi) = \mathbf{t}$. At the same time, $(\mathcal{I}',\mathcal{J}') \models_3^{\boldsymbol{\mathcal{U}}} \psi$ implies that $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$. Or equivalently, $(\mathcal{I},\mathcal{J}) \not\models_3^{\boldsymbol{\mathcal{U}}} \psi$ implies $(\mathcal{I}',\mathcal{J}') \not\models_3^{\boldsymbol{\mathcal{U}}} \psi$. Consequently $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) = \mathbf{f}$ implies $\mathcal{H}^3_{(\mathcal{I}',\mathcal{J}')}(\psi) = \mathbf{f}$. If $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) = \mathbf{u}$ no restrictions are imposed on $\mathcal{H}^3_{(\mathcal{I}',\mathcal{J}')}(\psi)$. In all three scenarios it holds that if $(\mathcal{I},\mathcal{J}) \leq_p (\mathcal{I}',\mathcal{J}')$, then $\mathcal{H}^3_{(\mathcal{I},\mathcal{J})}(\psi) \leq_p \mathcal{H}^3_{(\mathcal{I}',\mathcal{J}')}(\psi)$.

□

Taking $\mathfrak{L}$ as FO and $\mathcal{H}^3$ as the strong Kleene truth assignment, it is a folk result that $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ if ψ evaluates to true when interpreting all positively occurring atoms in $\mathcal{I}$ and all negatively occurring ones in $\mathcal{J}$. For $(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$, exchange the roles of $\mathcal{I}$ and $\mathcal{J}$. Now we can use aforementioned *TSRs* and their relation to the three-valued truth-function to define an associated approximator on ${\boldsymbol{\mathcal{E}}_{\boldsymbol{P}}}^c$.

Definition 4.3.5. *Let P be an $\mathfrak{L}$-program, and let $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$ be lower- and upper-regular TSRs. The approximator A_P induced by $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$ is defined such that for very $(\mathcal{I},\mathcal{J}) \in {\boldsymbol{\mathcal{E}}_{\boldsymbol{P}}}^c$: $A_P(\mathcal{I},\mathcal{J}) = (A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I},\mathcal{J}), A_P^{\models_3^{\boldsymbol{\mathcal{U}}}}(\mathcal{I},\mathcal{J}))$ with*

- $A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I},\mathcal{J}) = \{p \mid \exists (p \leftarrow \psi) \in P : (\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi\}$, *and*
- $A_P^{\models_3^{\boldsymbol{\mathcal{U}}}}(\mathcal{I},\mathcal{J}) = \{p \mid \exists (p \leftarrow \psi) \in P : (\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{U}}} \psi\}$

Proposition 4.3.6. *If $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$ are lower- and upper-regular TSRs, then A_P is an $\boldsymbol{\mathcal{E}_P}^c$-approximator.*

Proof. We start by proving $\leq_p$-monotonicity. Assume $(\mathcal{I},\mathcal{J}) \leq_p (\mathcal{I}',\mathcal{J}')$, as $\models_3^{\boldsymbol{\mathcal{L}}}$ is lower-regular, we find for every formula ψ in $\mathcal{L}$

$$(\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi \implies (\mathcal{I}',\mathcal{J}') \models_3^{\boldsymbol{\mathcal{L}}} \psi$$

and thus

$$\exists(p \leftarrow \psi) \in P : (\mathcal{I},\mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi \implies \exists(p \leftarrow \psi) \in P : (\mathcal{I}',\mathcal{J}') \models_3^{\boldsymbol{\mathcal{L}}} \psi$$

which implies $p \in A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I},\mathcal{J}) \implies p \in A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I}',\mathcal{J}')$, and thus

$$A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I},\mathcal{J}) \subseteq A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I}',\mathcal{J}').$$

Analogously, we find that $A_P^{\models_3^{\boldsymbol{\mathcal{U}}}}(\mathcal{I},\mathcal{J}) \supseteq A_P^{\models_3^{\boldsymbol{\mathcal{U}}}}(\mathcal{I}',\mathcal{J}')$. Hence $A_P(\mathcal{I},\mathcal{J}) \leq_p A_P(\mathcal{I}',\mathcal{J}')$. Thus, A_P is $\leq_p$-monotonic. Furthermore, we need to show that A_P extends T_P. By definition $T_P(\mathcal{I}) = \{p \mid p \leftarrow \psi \in P, \mathcal{I} \models_2 \psi\}$. As $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$ are *TSRs*, we know that for every formula ψ over $\mathcal{L}$, $(\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ iff $\mathcal{I} \models_2 \psi$ iff $(\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{U}}} \psi$. Therefore,

$$\begin{aligned} A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I},\mathcal{I}) &= \{p \mid \exists(p \leftarrow \psi) \in P : (\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{L}}} \psi\} \\ &= T_P(\mathcal{I}) = \{p \mid \exists(p \leftarrow \psi) \in P : \mathcal{I} \models_2 \psi\} \\ = A_P^{\models_3^{\boldsymbol{\mathcal{U}}}}(\mathcal{I},\mathcal{I}) &= \{p \mid \exists(p \leftarrow \psi) \in P : (\mathcal{I},\mathcal{I}) \models_3^{\boldsymbol{\mathcal{U}}} \psi\}. \end{aligned}$$

Hence $A_P(\mathcal{I},\mathcal{I}) = (T_P(\mathcal{I}), T_P(\mathcal{I}))$. In conclusion, $A_P(\mathcal{I},\mathcal{J})$ is an approximating operator. □

Proposition 4.3.7. *If $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$ are lower- and upper-regular TSRs, then A_P is isomorphic to the three-valued Φ_P induced by the three-valued truth assignment $\mathcal{H}^3$ combining $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$.*

Proof. Let $\mathfrak{A}$ be a three-valued interpretation that is represented by a pair of two-valued interpretations $(\mathcal{I},\mathcal{J})$. Similarly, let $\Phi_P(\mathfrak{A})$ be represented by a pair of two-valued interpretations $(\Phi_P(\mathfrak{A})_1, \Phi_P(\mathfrak{A})_2)$. Note that then, by definition of Φ_P

$$\Phi_P(\mathfrak{A})_1 = \left\{p \in \Sigma \middle| \mathrm{lub}_{\leq}(\{\mathcal{H}^3(\mathfrak{A})(\psi) \mid (p \leftarrow \psi) \in P\}) = \mathbf{t}\right\}$$

$$\Phi_P(\mathfrak{A})_2 = \left\{p \in \Sigma \middle| \mathrm{lub}_{\leq}(\{\mathcal{H}^3(\mathfrak{A})(\psi) \mid (p \leftarrow \psi) \in P\}) \in \{\mathbf{t},\mathbf{u}\}\right\}$$

As $\mathbf{t}$ is the $\leq$-greatest element, $\mathrm{lub}_{\leq}(\{\mathcal{H}^3(\mathfrak{A})(\psi) \mid (p \leftarrow \psi) \in P\}) = \mathbf{t}$ iff there exists a rule $(p \leftarrow \psi) \in P$ such that $\mathcal{H}^3(\mathfrak{A})(\psi) = \mathbf{t}$. Moreover, $\mathcal{H}^3(\mathfrak{A})(\psi) = \mathbf{t}$ iff $(\mathcal{I}, \mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$. Therefore

$$\begin{aligned}\Phi_P(\mathfrak{A})_1 &= \big\{p \in \Sigma \big| \mathrm{lub}_{\leq}(\{\mathcal{H}^3(\mathfrak{A})(\psi) \mid (p \leftarrow \psi) \in P\}) = \mathbf{t}\big\} \\ &= \{p \in \Sigma \mid \exists (p \leftarrow \psi) \in P : (\mathcal{I}, \mathcal{J}) \models_3^{\boldsymbol{\mathcal{L}}} \psi\} \\ &= A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I}, \mathcal{J})\end{aligned}$$

Analogously, we find $\Phi_P(\mathfrak{A})_2 = A_P^{\models_3^{\boldsymbol{\mathcal{U}}}}(\mathcal{I}, \mathcal{J})$. Therefore Φ_P is isomorphic to A_P. $\square$

Now *KK*, supported, *WF* and stable models of the $\mathscr{L}$-programs P can be defined in terms of $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$. Interestingly, using the semi-constructive approach to finding stable models, $\mathcal{J}$ is a stable model of P iff $\mathcal{J}$ is the least fixpoint of $\lambda \mathcal{I} \in [\bot, \mathcal{J}] : A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I}, \mathcal{J})$. This suggests that in the context of ASP where answer sets correspond to stable models, there is no need for $\models_3^{\boldsymbol{\mathcal{U}}}$! Clearly there exists an asymmetry between truth and falsity in stable models. While information about the truth of formulas, encoded by $\models_3^{\boldsymbol{\mathcal{L}}}$, is essential to determine the stable fixpoints of a program, information about their falsity, given by $\models_3^{\boldsymbol{\mathcal{U}}}$, is disregarded.

4.4 Generalizing the concept of answer set

Here, we use the notion of stable models of $\mathscr{L}$-programs to generalize answer sets. Now, $\mathscr{L}$, the logic of the rule bodies, has, besides a satisfaction relation $\models_2$ also a lower-regular *TSR* $\models_3^{\boldsymbol{\mathcal{L}}}$.

Definition 4.4.1. *$\mathcal{I}$ is an answer set of P if*

- *for every $(p \leftarrow \psi) \in P$, if $\mathcal{I} \models_2 \psi$ then $\mathcal{I} \models_2 p$, and*
- *there is no $\mathcal{J} \subsetneq \mathcal{I}$ such that for every $(p \leftarrow \psi) \in P$, if $(\mathcal{J}, \mathcal{I}) \models_3^{\boldsymbol{\mathcal{L}}} \psi$ then $(\mathcal{J}, \mathcal{I}) \models_3^{\boldsymbol{\mathcal{L}}} p$.*

Proposition 4.4.2 (semi-constructive answer sets)**.** *If $\models_3^{\boldsymbol{\mathcal{L}}}$ is lower-monotonic, then for $\mathscr{L}$-programs P, $\mathcal{I}$ is an answer set of P iff $\mathcal{I}$ is the limit of the increasing sequence $(\mathcal{I}_i)_{i \leq \beta}$ where*

- $\mathcal{I}_0 = \emptyset$,

- $\mathcal{I}_{i+1} = A_P^{\models_3^{\mathcal{L}}}(\mathcal{I}_i, \mathcal{I})$, *and*
- $\mathcal{I}_\lambda = \bigcup_{i<\lambda} \mathcal{I}_i$ *for limit ordinal* λ.

Proof. If $\models_3^{\mathcal{L}}$ is lower-monotonic, then for $\mathcal{L}$-programs P we find that $\lambda\mathcal{J} \in [\bot, \mathcal{I}] : A_P^{\models_3^{\mathcal{L}}}(\mathcal{J}, \mathcal{I}) = \lambda\mathcal{J} \in [\bot, \mathcal{I}] : \{p \mid \exists (p \leftarrow \psi) \in P : (\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} \psi\}$ is a monotonic operator. By the constructive version of the Knaster–Tarski theorem for monotonic operators, we know that the described increasing sequence converges to the least fixpoint of the operator. Therefore, all we need to proof is that the least fixpoint of $\lambda\mathcal{J} : A_P^{\models_3^{\mathcal{L}}}(\mathcal{J}, \mathcal{I})$ is $\mathcal{I}$ iff $\mathcal{I}$ is an answer set of P. $\mathcal{I}$ is the least fixpoint of $\lambda\mathcal{J} : A_P^{\models_3^{\mathcal{L}}}(\mathcal{J}, \mathcal{I})$ iff (i) $A_P^{\models_3^{\mathcal{L}}}(\mathcal{I}, \mathcal{I}) = \mathcal{I}$ and (ii) there is no $\mathcal{J} \subsetneq \mathcal{I}$ such that $A_P^{\models_3^{\mathcal{L}}}(\mathcal{J}, \mathcal{I}) = \mathcal{J}$.

1. (i) and (ii) $\implies$ $\mathcal{I}$ is an answer set.
   If $A_P^{\models_3^{\mathcal{L}}}(\mathcal{J}, \mathcal{I}) = \mathcal{J}$, then for every rule $p \leftarrow \psi \in P$, it holds that if $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} \psi$, then $p \in \mathcal{J}$ and thus $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} p$. Thus if (i), then for every $(p \leftarrow \psi) \in P$, if $(\mathcal{I}, \mathcal{I}) \models_3^{\mathcal{L}} \psi$ and hence $\mathcal{I} \models_2 \psi$, then $(\mathcal{I}, \mathcal{I}) \models_3^{\mathcal{L}} p$ and hence $\mathcal{I} \models_2 p$. This is the first condition of Definition 4.4.1.

   As $\lambda\mathcal{J} : A_P^{\models_3^{\mathcal{L}}}(\mathcal{J}, \mathcal{I})$ is monotonic and $\mathcal{I}$ is the least fixpoint, we find that from (ii), it follows that for every $\mathcal{J} \subsetneq \mathcal{I}$, there exists a rule $(p \leftarrow \psi) \in P$, such that $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} \psi$ and $(\mathcal{J}, \mathcal{I}) \not\models_3^{\mathcal{L}} p$, i.e., there is no $\mathcal{J} \subsetneq \mathcal{I}$ such that for every rule $(p \leftarrow \psi) \in P$, if $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} \psi$ then $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} p$. This is the second condition of Definition 4.4.1, so $\mathcal{I}$ is an answer set.

2. $\mathcal{I}$ is an answer set $\implies$ (i).
   If $\mathcal{I}$ is an answer set, then for every $(p \leftarrow \psi) \in P$, if $\mathcal{I} \models_2 \psi$, then $\mathcal{I} \models_2 p$. By definition of $A_P^{\models_3^{\mathcal{L}}}$, this entails that $A_P^{\models_3^{\mathcal{L}}}(\mathcal{I}, \mathcal{I}) = \mathcal{J} \subseteq \mathcal{I}$. Assume $\mathcal{J} \subsetneq \mathcal{I}$, then for every rule $p \leftarrow \psi \in P$, if $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} \psi$, then by lower-monotonicity of $\models_3^{\mathcal{L}}$, $(\mathcal{I}, \mathcal{I}) \models_3^{\mathcal{L}} \psi$, hence by definition of $A_P^{\models_3^{\mathcal{L}}}$, $p \in A_P^{\models_3^{\mathcal{L}}}(\mathcal{I}, \mathcal{I}) = \mathcal{J}$, which entails $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} p$. Therefore $\mathcal{J}$ violates the second condition of Definition 4.4.1.

3. $\mathcal{I}$ is an answer set $\implies$ (ii).
   Assume that there exists a $\mathcal{J} \subsetneq \mathcal{I}$, such that $A_P^{\models_3^{\mathcal{L}}}(\mathcal{J}, \mathcal{I}) = \mathcal{J}$, then it holds that for every rule $(p \leftarrow \psi) \in P$, if $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} \psi$, then $(\mathcal{J}, \mathcal{I}) \models_3^{\mathcal{L}} p$, so $\mathcal{J}$ violates the second condition of Definition 4.4.1.

□

In general, the (transfinite) fixpoint sequence may leave $[\bot, \mathcal{I}]$, or end up with a fixpoint $\mathcal{J} \subseteq \mathcal{I}$. In case it is $\mathcal{I}$, it is an answer set. Let $\models_3^{\boldsymbol{\mathcal{L}}}$ be lower-regular. Combining it with *any* upper-regular *TSR* $\models_3^{\boldsymbol{\mathcal{U}}}$ induces a regular truth assignment $\mathcal{H}^3$, as well as an entire family of AFT fixpoints and models of $\mathcal{L}$-programs: KK-models, WF-models and AFT-stable models approximated by the WF-models. The AFT-stable models in this framework depend only on $\models_3^{\boldsymbol{\mathcal{L}}}$ and they correspond exactly to answer sets of $\models_3^{\boldsymbol{\mathcal{L}}}$, since a lower-regular *TSR* is also lower-monotonic.

Proposition 4.4.3. *Answer sets and AFT-stable models coincide for $\mathcal{L}$-programs with a lower-regular TSR.*

Proof. If $\models_3^{\boldsymbol{\mathcal{L}}}$ is lower-regular, then the AFT-stable models are defined. Recall from Section 2 that, according to the constructive test, x is a stable model iff x is the limit of the lfp construction of $\lambda z : A^1(z, x)$ with A an approximator on $\boldsymbol{\mathcal{E}}^c$. From Proposition 4.3.6, we know that $A_P(\mathcal{I}, \mathcal{J}) = (A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I}, \mathcal{J}), A_P^{\models_3^{\boldsymbol{\mathcal{U}}}}(\mathcal{I}, \mathcal{J}))$ is an approximator for T_P if $\models_3^{\boldsymbol{\mathcal{L}}}$ and $\models_3^{\boldsymbol{\mathcal{U}}}$ are lower- and upper-regular. This holds as long as we arbitrarily choose an upper-regular $\models_3^{\boldsymbol{\mathcal{U}}}$. Then the first component corresponds to $A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I}, \mathcal{J})$. Thus $\mathcal{J}$ is a stable model iff $\mathcal{J}$ is the limit of the lfp construction of $\lambda\mathcal{I} : A_P^{\models_3^{\boldsymbol{\mathcal{L}}}}(\mathcal{I}, \mathcal{J})$ iff $\mathcal{J}$ is an answer set (by Proposition 4.4.2). □

To finish this section, we analyze the link with the original definition of answer set given by Gelfond and Lifschitz [46]. There, $\mathcal{L}$ is the logic of ground sets/conjunctions of literals with FO's standard satisfaction relation $\models_2$. Let P be an $\mathcal{L}$-program.

Definition 4.4.4 (Gelfond-Lifschitz reduct and answer set)**.** *The Gelfond-Lifschitz reduct (defined by Gelfond and Lifschitz [46]) $P^{\mathcal{J}}$ of P for an interpretation $\mathcal{J}$ is obtained from P by deleting*

- *all rules with a negative literal $\neg l$ such that $l \in \mathcal{J}$.*
- *all negative literals in the bodies of the remaining rules.*

$\mathcal{J}$ is a GL-answer set of P if $\mathcal{J} \models_2 P^{\mathcal{J}}$ and there is no $\mathcal{I} \subsetneq \mathcal{J}$ such that $\mathcal{I} \models_2 P^{\mathcal{J}}$.

We now identify the lower-regular *TSR* $\models_{GL}$ such that answer sets of programs induced by $\models_{GL}$, coincide with *GL*-answer sets. The *TSR* that is needed, evaluates bodies ψ in pairs $(\mathcal{I}, \mathcal{J})$ by interpreting atomic literals of ψ in $\mathcal{I}$ and negative literals in $\mathcal{J}$. But as explained in the previous section, that is how the lower-regular *TSR* $\models_{SK}$ of the strong Kleene truth assignment operates: $(\mathcal{I}, \mathcal{J}) \models_{SK} \psi$ iff atoms in ψ hold in $\mathcal{I}$ and negative literals hold in $\mathcal{J}$. Thus, $\models_{GL}$ is the restriction of $\models_{SK}$ to conjunctions of literals. With this in mind, the following propositions are straightforward.

Proposition 4.4.5. *For $\mathcal{I} \in [\bot, \mathcal{J}]$, $\mathcal{I} \models_2 P^{\mathcal{J}}$ iff for every rule $p \leftarrow \psi \in P$, if $(\mathcal{I}, \mathcal{J}) \models_{GL} \psi$ then $(\mathcal{I}, \mathcal{J}) \models_{GL} p$.*

Proof. For every rule $r^{\mathcal{J}}$ in $P^{\mathcal{J}}$ there exists a counterpart r in P, however, the opposite is not true. Since the head of a rule is never changed in the reduct, proving equivalence of the relations corresponds to showing for every rule $p \leftarrow \psi$ that $(\mathcal{I}, \mathcal{J}) \models_{GL} \psi$ if and only if $(p \leftarrow \psi)^{\mathcal{J}}$ exists and $\mathcal{I} \models_2 \psi^{\mathcal{J}}$.

Any rule $p \leftarrow \psi$ in P is of the form $p \leftarrow l_1 \wedge ... \wedge l_n \wedge \neg t_1 \wedge ... \wedge \neg t_m$ with $l_1, ..., l_n, t_1, ..., t_m$ atoms. If there exists an atom t_i such that $t_i \in \mathcal{J}$, then the rule is deleted during the first step of the construction of the reduct. Consequently, for every $\mathcal{I}$ we expect $(\mathcal{I}, \mathcal{J}) \not\models_{GL} \psi$. Indeed, from $t_i \in \mathcal{J}$ the satisfaction relation deduces that $(\mathcal{I}, \mathcal{J}) \not\models_{GL} \neg t_i$ and therefore $(\mathcal{I}, \mathcal{J}) \not\models_{GL} \psi$.

If for every $j \in [1, m]$ it holds that $t_j \notin \mathcal{J}$, then $p \leftarrow \psi$ is transformed to $(p \leftarrow \psi)^{\mathcal{J}}$ during the second step of the reduct-construction. This new rule is now given by $p \leftarrow l_1 \wedge ... \wedge l_n$. From the definition of the satisfaction relation $\models_2$ we know that $\mathcal{I} \models_2 \psi^{\mathcal{J}}$ if and only if $l_1, ..., l_n \in \mathcal{I}$ and because $(\mathcal{I}, \mathcal{J}) \models_{GL} \neg t_i$ for all i, we have $(\mathcal{I}, \mathcal{J}) \models_{GL} \psi$. Hence $\mathcal{I} \models_2 \psi^{\mathcal{J}}$ iff $(\mathcal{I}, \mathcal{J}) \models_{GL} \psi$. □

Proposition 4.4.6. *$\mathcal{J}$ is a GL-answer set of P iff $\mathcal{J}$ is an AFT-stable model of P under strong Kleene truth assignment.*

Proof. We now know that for $\mathcal{I} \in [\bot, \mathcal{J}]$, $\mathcal{I} \models_2 P^{\mathcal{J}}$ iff $(\mathcal{I}, \mathcal{J}) \models_{GL} P$ with $P^{\mathcal{J}}$ the *GL*-reduct of P for $\mathcal{J}$. Since $\models_{GL}$ is lower-regular, we know that $\mathcal{J}$ is an AFT stable model iff $\mathcal{J}$ is an answer set of P by Proposition Proposition 4.4.3. $\mathcal{J}$ is an answer set of P iff $\mathcal{J} \models_2 P$ and there is no $\mathcal{I} \subsetneq \mathcal{J}$ such that $(\mathcal{I}, \mathcal{J}) \models_{GL} P$. Hence, $\mathcal{J}$ is an answer set of P iff $\mathcal{J} \models_2 P$ and there is no $\mathcal{I} \subsetneq \mathcal{J}$ such that $\mathcal{I} \models_2 P^{\mathcal{J}}$ iff $\mathcal{J}$ is a *GL*-answer set. □

Thus, this type of answer set fits in the AFT-landscape of KK, WF and stable models.

4.5 Aggregate Programs in the AFT Framework

Aggregate Programs For the remainder of the text we will consider $\mathfrak{L}$-programs where $\mathfrak{L}$ is the logic of conjunctions of literals and positive aggregate atoms.[2]

An aggregate atom a^{Aggr} is of the form: $Agg(\{a_1 : cond_1, ..., a_n : cond_n\}) * w$ with aggregate symbol Agg (e.g., SUM), comparison connective $*$ (e.g., $\leq, =, \neq, \dots$),

[2] In many semantics, including AFT semantics, a negated aggregate atom can always be represented by its dual positive aggregate atom. E.g., $\neg(SUM(\{1 : s\}) > 0)$ corresponds to $SUM(\{1 : s\}) \leq 0$.

numerical value w and multi-set $\{a_1 : cond_1, ..., a_n : cond_n\}$ where each $cond_i$ is a literal and each a_i is a weight. The $\models_2$ relation for $\mathcal{L}$ is naturally extended with a rule for evaluating aggregate atoms, so T_P is also defined.

Example 4.5.1. *Some common examples of aggregate symbols and their informal semantics: if $S \subseteq \mathbb{N}$, then*

- *$SUM(S)$ denotes the sum $\sum S$.*
- *$PROD(S)$ denotes the product $\prod S$.*
- *$CARD(S)$ denotes the cardinality of S, i.e., $|S|$.*
- *$MIN(S)$ denotes the minimum of S.*
- *$MAX(S)$ denotes the maximum of S.*

For example, consider $a^{Aggr} = SUM(\{-1 : p, 1 : q\}) \leq 0$. Then $\{p, q\} \models_2 a^{Aggr}$ as $-1+1 = 0$. Similarly, $\{p\} \models_2 a^{Aggr}$ and $\{\} \models_2 a^{Aggr}$. However, $\{q\} \not\models_2 a^{Aggr}$ as $1 \not\leq 0$.

This enables the first approach to applying AFT on this type of program, using the three-valued ultimate approximator U_{T_P} yielding the ultimate KK, WF and stable semantics. Up to the syntax, this is the semantics of Denecker et al. [34]. The second approach is based on defining a regular three-valued truth assignment for aggregate atoms, leading to a three valued operator Φ_P^{Aggr}. Up to the syntax, this was the approach followed by Pelov et al. [69].

We now start the study of existing approaches for handling aggregates in ASP. Some of these allow more extensive programs than the ones analyzed here; however our analysis only considers the simpler $\mathcal{L}$-programs. Proposition 4.4.2 shows that there is a constructive test for answer sets of non-disjunctive $\mathcal{L}$-programs for semantics with lower-monotonic *TSR*'s. A lower-regular *TSR* is lower-monotonic by definition. Thus, this constructive test is applicable for all semantics that fit in the AFT-framework. However, not all semantics for ASP-programs in the literature have lower-regular or even lower-monotonic ternary satisfaction relations.

Before we start, we define the precision order on the *TSRs* analogous to the precision order on truth assignments defined by Pelov et al. [69].

Definition 4.5.2 (Precision relation over lower ternary satisfaction relations)**.** *A TSR $\models_a$ is less precise than a TSR $\models_b$, or $\models_a \leq_p \models_b$, if for every formula ψ and every pair of two-valued interpretations $(\mathcal{I}, \mathcal{J})$: $(\mathcal{I}, \mathcal{J}) \models_a \psi$ implies $(\mathcal{I}, \mathcal{J}) \models_b \psi$.*

Proposition 4.5.3. *Let $\models_a, \models_b$ be TSRs that coincide with $\models_{GL}$ on aggregate free bodies. If $\models_a \leq_p \models_b$ and $\mathcal{J}$ is an answer set associated with $\models_a$ (an a-answer set), then $\mathcal{J}$ is a b-answer set.*

Proof. Assume that $\mathcal{J}$ is an a-answer set. This means that

(i) for every $(p \leftarrow \psi) \in P$, if $\mathcal{J} \models_2 \psi$, then $\mathcal{J} \models_2 p$, and

(ii) for every $\mathcal{I} \subsetneq \mathcal{J}$ it holds that there exists a rule $(p \leftarrow \psi) \in P$ such that $(\mathcal{I}, \mathcal{J}) \models_a \psi$ and $(\mathcal{I}, \mathcal{J}) \not\models_a p$

Observe that (i) also holds for $\models_b$ as it does not depend on $\models_a$; it remains to show that (ii) also holds for $\models_b$. By $\models_a \leq_p \models_b$, we find that $(\mathcal{I}, \mathcal{J}) \models_b \psi$. Moreover, in the considered programs, p can only be a propositional atom. If $(\mathcal{I}, \mathcal{J}) \not\models_a p$, then $p \notin \mathcal{I}$ and thus $(\mathcal{I}, \mathcal{J}) \not\models_b p$. Consequently, $\mathcal{J}$ is a b-answer set of P. □

4.5.1 Approaches that fit in the AFT framework

Pelov et al. [69] introduces several regular truth assignments for aggregate atoms. This is equivalent with expanding the lower- and upper-regular *TSR* to a rule for aggregate atoms. As argued before, for defining answer sets, the lower-regular *TSR* suffices.[3] The least precise approximation, $\mathcal{H}^{triv}$ assigns to an aggregate atom a^{Aggr} the same value as $\mathcal{I}$ and $\mathcal{J}$ when $\mathcal{I}$ and $\mathcal{J}$ agree on the conditions in a^{Aggr} and **u** when they disagree. The corresponding lower *TSR* is:

Definition 4.5.4 ($\models_{triv}$). *$\models_{triv}$ extends $\models_{GL}$ with: $(\mathcal{I}, \mathcal{J}) \models_{triv} a^{Aggr}$ iff $\mathcal{J} \models_2 a^{Aggr}$ and $cond_i \in \mathcal{I}$ iff $cond_i \in \mathcal{J}$ for every condition $cond_i$ in a^{Aggr}.*

The most precise regular truth assignment $\mathcal{H}^{ult}$ assigns **t** (**f**) to an aggregate expression in $(\mathcal{I}, \mathcal{J})$ if it is **t** (**f**) in every $\mathcal{I}' \in [\mathcal{I}, \mathcal{J}]$. Otherwise, it assigns **u**. The corresponding lower *TSR* is:

Definition 4.5.5 ($\models_{ult}$). *$\models_{ult}$ extends $\models_{GL}$ with: $(\mathcal{I}, \mathcal{J}) \models_{ult} a^{Aggr}$ iff for each $\mathcal{I}'$ such that $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$: $\mathcal{I}' \models_2 a^{Aggr}$.*

Pelov [67] shows that for stratified aggregate programs (where any propositional atoms in aggregate expressions are fully defined at a lower level), the trivial and the ultimate truth assignments lead to the same semantics.

[3] The formalism of Pelov et al. [69] is much richer including aggregate atoms under negation, and its semantics requires lower- and upper-regular *TSRs* defined inductively in terms of each other.

While very precise, according to Pelov [67], the complexity of computing KK, WF and stable models under $\mathcal{H}^{ult}$ moves to the next level of the polynomial hierarchy. To avoid this, Pelov offers a less precise alternative called the *bounded* truth assignment $\mathcal{H}^{bnd}$.

Phrased in terms of the present aggregate programs, it uses functions $LB_{Agg}, UB_{Agg} : 2^{\tilde{D}_1} \to D_2$ that map any three-valued multi-set $\{ms\}^{(\mathcal{I},\mathcal{J})}$ to respectively the minimum and the maximum of $\{Agg(\{ms\})^{\mathcal{I}'} \mid \mathcal{I}' \in [\mathcal{I},\mathcal{J}]\}$; i.e., LB_{Agg} represents the lower bound for the aggregate function Agg on the possible multi-sets and UB_{Agg} the upper bound. Then $\mathcal{H}^{bnd}_{(\mathcal{I},\mathcal{J})}(Agg(\{ms\}) * w) = \mathbf{t}$ if for all $n \in [LB_{Agg}, UB_{Agg}] : n * w$, and $\mathcal{H}^{bnd}_{(\mathcal{I},\mathcal{J})}(Agg(\{ms\}) * w) = \mathbf{f}$ if for all $n \in [LB_{Agg}, UB_{Agg}]$, not $n * w$. Interestingly, when $\mathcal{H}^{bnd}$ is split in a lower and upper *TSR*, the corresponding lower *TSR* $\models_{bnd}$ coincides with $\models_{ult}$. The real difference between $\mathcal{H}^{bnd}$ and $\mathcal{H}^{ult}$ is captured by the upper *TSR*.

Example 4.5.6. *For example, if* $a^{Aggr} = SUM(\{1 : p, 0 : q, -1 : r\}) = 0$ *with* $\Sigma = \{p, q, r\}$*, then, e.g.,*

- $(\{\}, \{q\}) \models_{triv} a^{Aggr}$,
- $(\{p, r\}, \{p, q, r\}) \not\models_{triv} a^{Aggr}$, *and*
- $(\{p, r\}, \{p, q, r\}) \models_{ult} a^{Aggr}$

Pelov [67] lists polynomial algorithms to compute both bounds for all aggregate atoms discussed in his thesis. The same holds for the common aggregate atoms in ASP. Consequently, the complexity of computing the different types of models remains on the same level as for the non-aggregate case.

Proposition 4.5.7. *The TSRs* $\models_{triv}$ *and* $\models_{ult}$ *(*$\models_{bnd}$*) are lower-regular. Since* $\models_{triv} \leq_p \models_{ult}$*, any answer set of* $\models_{triv}$ *is one of* $\models_{ult}$*.*

Proof. From definition 7.1 in the paper by Pelov et al. [68], it follows that $\mathcal{H}^{triv}$ and $\mathcal{H}^{ult}$ are regular. Then the first part of this proposition follows from Proposition 4.3.4, the second part from Proposition 4.5.3. □

Liu et al. [57] introduced a kind of *TSR* to define semantics for abstract constraints. While the restrictions imposed on these relations are different and do not necessarily fit into AFT, their main example, the sub-satisfiability relation as proposed by Son et al. [81] does. For any abstract constraint α: $(\mathcal{I},\mathcal{J}) \models_{LPST} \alpha$ if and only if for each interpretation $\mathcal{I}'$ such that $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$, it holds that $\mathcal{I}' \models_2 \alpha$.

Definition 4.5.8 ($\models_{LPST}$)**.** *The TSR $\models_{LPST}$ extends $\models_{GL}$ with: If a^{Aggr} is an aggregate atom, then $(\mathcal{I},\mathcal{J}) \models_{LPST} a^{Aggr}$ iff for each $\mathcal{I}'$ such that $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$: $\mathcal{I}' \models_2 a^{Aggr}$.*

This is the same satisfaction relation as $\models_{ult}$ in Definition 4.5.5; hence it is lower-regular and defines the same answer sets.

Gelfond and Zhang [47] construct a reduct for aggregate programs with respect to a three-valued interpretation. We only consider the case where the interpretation is two-valued. Gelfond and Zhang [47] allow two kinds of negation: *negation by default*, which corresponds to negation as presented in this paper, and *explicit negation*, which is not a part of the syntax of the programs considered here but can be simulated by the well-known translation of explicitly negated atoms of a predicate p into atoms of a newly introduced predicate p^*. Since Gelfond and Zhang [47] do not allow default negation within an aggregate atom, here it suffices to consider programs with positive conditions inside an aggregate atom. The reduction process is split into two main phases. The first phase constructs a reduct regarding the aggregate atoms. It consists of two steps:

1. Removing all rules with aggregate atoms that evaluate to **f** in the interpretation.
2. Replacing every remaining aggregate atom by the conjunction of the subset of its conditions that are **t** in the interpretation.

In other words, given a rule r in a program P: $p \leftarrow l_1 \wedge ... \wedge l_n$, such that for an l_i it holds that $l_i = Agg(\{a_1 : cond_1, ..., a_n : cond_n\}) * w$, then the rule is deleted in the reduct $P^{\mathcal{J}}$ if l_i evaluates to **f** in $\mathcal{J}$. Otherwise the rule is replaced by $p \leftarrow l_1 \wedge ... \wedge l_{i-1} \wedge l_{i+1} \wedge ... \wedge l_n \wedge (\bigwedge\{cond_j \in \{cond_1, \ldots, cond_n\} \mid J \models_2 cond_j\}$.

The second phase transforms the preliminary reduct after the first phase to its Gelfond-Lifschitz reduct. In this way, it preserves the capability to deal with ordinary propositional atoms. From this reduct one can inductively define the *TSR* $\models_{GZ}$:

Definition 4.5.9 ($\models_{GZ}$)**.** *$\models_{GZ}$ extends $\models_{GL}$ with: Let $a^{Aggr} = Agg(\{a_1 : cond_1, ..., a_n : cond_n\}) * w$. $(\mathcal{I},\mathcal{J}) \models_{GZ} a^{Aggr}$ iff $\mathcal{J} \models_2 a^{Aggr}$ and $(\mathcal{I},\mathcal{J}) \models_{GZ} \bigwedge\{cond_j \in \{cond_1, \ldots, cond_n\} \mid \mathcal{J} \models_2 cond_j\}$.*

Proposition 4.5.10. *For aggregate programs containing only positive conditions in aggregate atoms, the TSR $\models_{GZ}$ is identical to the TSR $\models_{triv}$.*

Proof. We only need to proof this for aggregate atoms as $\models_{GZ}$ is truth-functional. We start by showing that for any $\mathcal{I} \subseteq \mathcal{J}$, $(\mathcal{I},\mathcal{J}) \models_{GZ} a^{Aggr}$ entails $(\mathcal{I},\mathcal{J}) \models_{triv} a^{Aggr}$. By definition of $\models_{GZ}$,

$(\mathcal{I},\mathcal{J}) \models_{GZ} a^{Aggr}$ iff $\mathcal{J} \models_2 a^{Aggr}$, and

$$(\mathcal{I},\mathcal{J}) \models_{GZ} \bigwedge\{cond_j \in \{cond_1,\ldots,cond_n\} \mid \mathcal{J} \models_2 cond_j\}.$$

The conditions are positive, so they are evaluated in $\mathcal{I}$, hence, $\mathcal{I} \models_2 \bigwedge\{cond_j \in \{cond_1,\ldots,cond_n\} \mid \mathcal{J} \models_2 cond_j\}$. Clearly, for the conditions in this set, $cond_i \in \mathcal{I}$ and $cond_i \in \mathcal{J}$. For the other conditions (such that $\mathcal{J} \not\models_2 cond_i$), they are false in $\mathcal{J}$, thus $cond_i \notin \mathcal{J}$ hence also $cond_i \notin \mathcal{I} \subseteq \mathcal{J}$. So, for all conditions, $cond_i \in \mathcal{I}$ iff $cond_i \in \mathcal{J}$ and $(\mathcal{I},\mathcal{J}) \models_{triv} a^{Aggr}$.

For the other direction, by definition of $\models_{triv}$, if $(\mathcal{I},\mathcal{J}) \models_{triv} a^{Aggr}$, then $\mathcal{J} \models_2 a^{Aggr}$ and $cond_i \in \mathcal{J}$ iff $cond_i \in \mathcal{I}$ for every condition $cond_i \in \{cond_1,\ldots,cond_n\}$. Therefore, for every condition $cond_i \in \{cond_j \in \{cond_1,\ldots,cond_n\} \mid \mathcal{J} \models_2 cond_j\}$ it holds that $\mathcal{I} \models_2 cond_i$. By definition of $\models_{GZ}$ for conjunction of literals, then $(\mathcal{I},\mathcal{J}) \models_{GZ} \bigwedge\{cond_j \in \{cond_1,\ldots,cond_n\} \mid \mathcal{J} \models_2 cond_j\}$. Hence, $(\mathcal{I},\mathcal{J}) \models_{GZ} a^{Aggr}$. □

Corollary 4.5.11. *$\models_{GZ}$ is lower-regular, i.e., $(\mathcal{I},\mathcal{J}) \models_{GZ} a^{Aggr}$ iff $(\mathcal{I},\mathcal{J}) \models_{triv} a^{Aggr}$.*

Precision Complexity Trade-off. One expects more effort gives more precise approximations. From theorem 7.4 in the paper by Pelov et al. [69], it follows that if the evaluation of an expression with respect to a lower-regular ternary satisfaction relation $\models_3^{\mathcal{L}}$ is polynomially computable, then checking whether or not a model is an answer set is in P and deciding whether an answer set for a program exists, is in NP. This is the case for $\models_{triv}$ and $\models_U$ and for instances of the framework that coincide with them. This includes all examples we have discussed up until now.

4.5.2 Other Ternary Satisfaction Relations

In the AFT-framework, semantics of ASP aggregate atoms are based on lower regular *TSRs*. As we show in this section, also other well-known semantics can be characterized using *TSRs*, however, they are not regular. This is no coincidence. The definition of answer set semantics in terms of *TSRs* strongly resembles another well-known semantic method of ASP, namely using the logic of here-and-there (**HT**) (for an overview of **HT** and its applications to ASP, see the work by Cabalar et al. [18]). Due to very different points of view on answer sets, the two frameworks obtain different requirements for the *TSRs*. AFT treats answer sets as the result of constructive processes; the rule operator serves to produce them. A production is safe if the *TSR* is lower-regular. In contrast, **HT** takes a non-constructive take on answer sets. In **HT**, extensions build on the inherently non-regular *TSRs* derived from the three-valued logic *G3* introduced by Gödel [48] where the rule operator is treated as **HT**-material implication.

Marek and Remmel [58] study Set Constraints (SC) Programming. It builds an *NSS-reduct* for an SC-program. Liu et al. [57] prove that this semantics for set

constraints is also obtained by the following satisfaction rule for a set constraint α: $(\mathcal{I}, \mathcal{J}) \models_{MR} \alpha$, if $\mathcal{J} \models_2 \alpha$ and there exists an interpretation $\mathcal{I}' \subseteq \mathcal{I}$ such that $\mathcal{I}' \models_2 \alpha$. This leads to the *TSR* $\models_{MR}$:

Definition 4.5.12 ($\models_{MR}$)**.** *$\models_{MR}$ extends $\models_{GL}$ with: $(\mathcal{I}, \mathcal{J}) \models_{MR} a^{Aggr}$ iff $\mathcal{J} \models_2 a^{Aggr}$ and there exists an interpretation $\mathcal{I}'$ such that $\mathcal{I}' \subseteq \mathcal{I}$ and $\mathcal{I}' \models_2 a^{Aggr}$.*

Consider the aggregate atom $SUM(\{1 : p, -1 : q\}) \geq 0$ and two intervals, namely $(\emptyset, \{p, q, s\}) <_p (\emptyset, \{q\})$. We have that $(\emptyset, \{p, q, s\}) \models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$ while $(\emptyset, \{q\}) \not\models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$. Hence, $\models_{MR}$ is not lower-regular.

Proposition 4.5.13. *$\models_{MR}$ extends $\models_2$, i.e., $(\mathcal{I}, \mathcal{I}) \models_{MR} \psi$ iff $\mathcal{I} \models_2 \psi$.*

Proof. The satisfaction relation is truth-functional hence we only need to prove this for aggregate atoms. We know that $(\mathcal{I}, \mathcal{I}) \models_{MR} a^{Aggr}$ iff $\mathcal{I} \models_2 a^{Aggr}$ and there exists an interpretation $\mathcal{I}' \subseteq \mathcal{I}$ such that $\mathcal{I}' \models_2 a^{Aggr}$. Take $\mathcal{I}' = \mathcal{I}$ and it is clear that $\mathcal{I} \models_2 a^{Aggr}$ implies that there exists a $\mathcal{I}$ that satisfies the requirements. This observation leads to $(\mathcal{I}, \mathcal{I}) \models_{MR} a^{Aggr}$ iff $\mathcal{I} \models_2 a^{Aggr}$. □

Proposition 4.5.14. *$\models_{MR}$ is lower-monotonic, i.e., if $\mathcal{I} \subseteq \mathcal{I}'$ and $(\mathcal{I}, \mathcal{J}) \models_{MR} \psi$, then $(\mathcal{I}', \mathcal{J}) \models_{MR} \psi$.*

Proof. The part of the definition for $\models_{MR}$ that was taken from $\models_{GL}$ satisfies this property since $(\mathcal{I}, \mathcal{J}) \leq_p (\mathcal{I}', \mathcal{J})$ and $\models_{GL}$ is lower-regular. For the rule regarding aggregates, it is clear that if $(\mathcal{I}, \mathcal{J}) \models_{MR} a^{Aggr}$, then $\mathcal{J} \models_2 a^{Aggr}$ and there exists a $\mathcal{J}' \subseteq \mathcal{I} \subseteq \mathcal{I}'$ such that $\mathcal{J}' \models_2 a^{Aggr}$. Clearly, this means that $(\mathcal{I}', \mathcal{J}) \models_{MR} a^{Aggr}$, so $\models_{MR}$ is lower-monotonic. □

While non-regular, the ternary relation $\models_{MR}$ still extends the satisfaction relation $\models_2$ and induces a monotonic operator $\lambda\mathcal{I} : A_{\models_{MR}}(\mathcal{I}, \mathcal{J})$. Thus, an answer set may still be defined as an interpretation $\mathcal{J}$ that is a least fixpoint of this operator. But, due to non-regularity, some unexpected answer sets are obtained.

Example 4.5.15. *Take the program:*

$$s \leftarrow SUM(\{1 : p, -1 : q\}) \geq 0.$$

$$q \leftarrow SUM(\{1 : s\}) > 0. \qquad p \leftarrow SUM(\{1 : q\}) > 0.$$

The bodies of these rules are equivalent to $p \vee \neg q$, respectively s and q. Therefore, one expects its answer sets to be the same as those of the simplified program:

$$s \leftarrow p. \qquad s \leftarrow \neg q. \qquad q \leftarrow s. \qquad p \leftarrow q.$$

To check that $\mathcal{J} = \{p, q, s\}$ is an answer set, we observe that $(\emptyset, \mathcal{J}) \models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$, hence the first iteration derives the head s. Two more iterations reconstruct the fixpoint $\{p, q, s\}$ which therefore is an answer set according to these semantics.

On the other hand, the Gelfond-Lifschitz reduct of the simplified program with respect to $\mathcal{J} = \{p, q, s\}$ is:

$$s \leftarrow p. \qquad q \leftarrow s. \qquad p \leftarrow q.$$

Since $\emptyset$ is a model of the reduct, $\{p, q, s\}$ is not an answer set of the simplified program. The culprit for having 'too many' answer sets of the aggregate program is the non-regularity of $\models_{MR}$ which leads to the *unsafe* derivation of aggregate atoms: in the first derivation, $(\emptyset, \{p, q, s\}) \models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$ which derived s, but this derivation was *unsafe* since this aggregate atom is not satisfied in the more precise $(\emptyset, \{q\})$. However, things change when looking only at convex aggregate atoms.

Definition 4.5.16 (Convex aggregate atom)**.** *An aggregate atom a^{Aggr} is convex iff for all interpretations $\mathcal{I}, \mathcal{I}', \mathcal{J}$, such that $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$, it holds that if $\mathcal{I} \models_2 a^{Aggr}$ and $\mathcal{J} \models_2 a^{Aggr}$, then $\mathcal{I}' \models_2 a^{Aggr}$.*

Proposition 4.5.17. *For convex aggregate atoms, $\models_{MR}$ behaves lower-regular.*

Proof. We have already shown that it extends the satisfaction relation $\models_2$ for arbitrary aggregate atoms, including convex ones. All that is left to prove is its $\leq_p$-monotonic behavior. If $(\mathcal{I}, \mathcal{J}) \leq_p (\mathcal{I}', \mathcal{J}')$ and $(\mathcal{I}, \mathcal{J}) \models_{MR} a^{Aggr}$ with a^{Aggr} a convex aggregate atom, then $\mathcal{J} \models_2 a^{Aggr}$ and there exists an interpretation $\mathcal{I}'' \subseteq \mathcal{I}$ such that $\mathcal{I}'' \models_2 a^{Aggr}$. According to the definition of convex aggregate atoms, this implies that for every interpretation $\mathcal{J}''$ such that $\mathcal{I}'' \subseteq \mathcal{J}'' \subseteq \mathcal{J}$, $\mathcal{J}'' \models_2 a^{Aggr}$. This includes the interpretations $\mathcal{I}, \mathcal{I}'$ and $\mathcal{J}'$. From this, it is trivial to see that $(\mathcal{I}', \mathcal{J}') \models_{MR} a^{Aggr}$. □

Proposition 4.5.18. *For convex aggregate atoms, $\models_{MR}$ behaves equivalent to $\models_{ult}$.*

Proof. If $(\mathcal{I}, \mathcal{J}) \models_{MR} a^{Aggr}$, then $\mathcal{J} \models_2 a^{Aggr}$ and there exists an interpretation $\mathcal{I}' \subseteq \mathcal{I}$ such that $\mathcal{I} \models_2 a^{Aggr}$. Since a^{Aggr} is convex, this means that for every $\mathcal{J}'$ such that $\mathcal{I}' \subseteq (\mathcal{I} \subseteq)\mathcal{J}' \subseteq \mathcal{J}$, $\mathcal{J}' \models_2 a^{Aggr}$. Thus, $(\mathcal{I}, \mathcal{J}) \models_{ult} a^{Aggr}$.

If $(\mathcal{I}, \mathcal{J}) \models_{ult} a^{Aggr}$, then $X \models_2 a^{Aggr}$ for every $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$, thus $\mathcal{J} \models_2 a^{Aggr}$ and $\mathcal{I} \models_2 a^{Aggr}$ with $\mathcal{I} \subseteq \mathcal{I}$. Therefore, $(\mathcal{I}, \mathcal{J}) \models_{MR} a^{Aggr}$. □

Faber et al. [41] provide semantics for a broader class of ASP programs including negated aggregate atoms. This approach constructs a reduct $P^{\mathcal{J}}$ for a program P with respect to an interpretation $\mathcal{J}$ by deleting all rules of which the body is not satisfied in $\mathcal{J}$. In general, the immediate consequence operator $T_{P^{\mathcal{J}}}$ is not monotonic, so answer

sets cannot be defined using the lfp construction of this operator. Nevertheless, the *FPL*-answer sets are the answer sets of the following *TSR*:

Definition 4.5.19 ($\models_{FPL}$)**.** *We define* $\models_{FPL}$ *as follows:* $(\mathcal{I},\mathcal{J}) \models_{FPL} \psi$ *iff* $\mathcal{I} \models_2 \psi$ *and* $\mathcal{J} \models_2 \psi$.

Proposition 4.5.20. $\models_{FPL}$ *extends* $\models_2$*, i.e.,* $(\mathcal{I},\mathcal{I}) \models_{FPL} \psi$ *iff* $\mathcal{I} \models_2 \psi$.

Proof. By definition $(\mathcal{I},\mathcal{I}) \models_{FPL} \psi$ iff $\mathcal{I} \models_2 \psi$ and $\mathcal{I} \models_2 \psi$. Hence $(\mathcal{I},\mathcal{I}) \models_{FPL} \psi$ iff $\mathcal{I} \models_2 \psi$. □

Proposition 4.5.21. *For conjunctions of aggregate free literals,* $\models_{FPL}$ *coincides with* $\models_{GL}$.

Proof. For literals l_i it is easy to see that: $(\mathcal{I},\mathcal{J}) \models_{FPL} l_i$ iff $(\mathcal{I},\mathcal{J}) \models_{GL} l_i$ since if $\mathcal{I} \models_2 p$, then $\mathcal{J} \models_2 p$ and if $\mathcal{J} \models_2 \neg p$, then $\mathcal{I} \models_2 \neg p$. For conjunctions of literals $\bigwedge_{i=1}^{n} l_i$, we have $(\mathcal{I},\mathcal{J}) \models_{FPL} \bigwedge_{i=1}^{n} l_i$ iff $\mathcal{I} \models_2 \bigwedge_{i=1}^{n} l_i$ and $\mathcal{J} \models_2 \bigwedge_{i=1}^{n} l_i$. By the definition of $\models_2$ for conjunctions, this is equivalent to $\mathcal{I} \models_2 l_i$ and $\mathcal{J} \models_2 l_i$ for every l_i in the conjunction. This in turn is equivalent to $(\mathcal{I},\mathcal{J}) \models_{FPL} l_i$. By means of the aforementioned result for literals, this is equivalent with $(\mathcal{I},\mathcal{J}) \models_{GL} l_i$. By definition of $\models_{GL}$ for conjunctions of literals, we then find $(\mathcal{I},\mathcal{J}) \models_{FPL} \bigwedge_{i=1}^{n} l_i$ iff $(\mathcal{I},\mathcal{J}) \models_{GL} \bigwedge_{i=1}^{n} l_i$. □

This very simple *TSR* is neither lower-regular nor lower-monotonic, thus the constructive test is inapplicable. This again easily illustrated by the aggregate atom $SUM(\{1:p,-1:q\}) \geq 0$. Clearly, $(\emptyset,\{p,q,s\}) \models_{FPL} SUM(\{1:p,-1:q\}) \geq 0$, while $(\{q\},\{p,q,s\}) \not\models_{FPL} SUM(\{1:p,-1:q\}) \geq 0$. As a consequence, discrepancies between some aggregate programs and their aggregate-free simplification are also present. Using again Example 4.5.15, similarly to $\models_{MR}$, the *FPL*-semantics leads to $\{p,q,s\}$ as an answer set. It is obvious that if $(\mathcal{I},\mathcal{J}) \models_{FPL} \psi$, then $(\mathcal{I},\mathcal{J}) \models_{MR} \psi$, since $\mathcal{I} \subseteq \mathcal{I}$. Accordingly $\models_{FPL}$ is less precise then $\models_{MR}$. Analogously, if $(\mathcal{I},\mathcal{J}) \models_{ult} \psi$, then $(\mathcal{I},\mathcal{J}) \models_{FPL} \psi$ since the interpretations $\mathcal{I}$ and $\mathcal{J}$ are both elements of $[I,J]$.

Proposition 4.5.22. *For convex aggregate atoms, the TSR* $\models_{FPL}$ *behaves lower-regular and equivalent to* $\models_{ult}$.

Proof.
- Firstly, we prove $\models_{ult} \leq_p \models_{FPL}$: If $(\mathcal{I},\mathcal{J}) \models_{ult} a^{Aggr}$, then $(\mathcal{I},\mathcal{J}) \models_{FPL} a^{Aggr}$.
  If $(\mathcal{I},\mathcal{J}) \models_{ult} a^{Aggr}$, then $\mathcal{I}' \models_2 a^{Aggr}$ for every $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$. Consequently, $\mathcal{I} \models_2 a^{Aggr}$ and $\mathcal{J} \models_2 a^{Aggr}$, hence $(\mathcal{I},\mathcal{J}) \models_{FPL} a^{Aggr}$.

- Secondly, we prove $\models_{FPL} \leq_p \models_{MR}$: If $(\mathcal{I},\mathcal{J}) \models_{FPL} a^{Aggr}$, then $(\mathcal{I},\mathcal{J}) \models_{MR} a^{Aggr}$.
  If $(\mathcal{I},\mathcal{J}) \models_{FPL} a^{Aggr}$, then $\mathcal{I} \models_2 a^{Aggr}$ and $\mathcal{J} \models_2 a^{Aggr}$. Thus $\mathcal{J} \models_2 a^{Aggr}$ and $\mathcal{I} \models_2 a^{Aggr}$ with $\mathcal{I} \subseteq \mathcal{I}$. Therefore, $(\mathcal{I},\mathcal{J}) \models_{MR} a^{Aggr}$.

This means that $\models_{ult} \leq_p \models_{FPL} \leq_p \models_{MR}$. However, for programs with only convex aggregates, $\models_{ult}$ and $\models_{MR}$ coincide. Therefore, $\models_{FPL}$ will also coincide with these semantics. Consequently, $\models_{FPL}$ is a lower-regular relation when we only consider convex aggregates. □

Ferraris [43]. This semantics is closely related to the *FPL*-semantics. Actually, they coincide for negation-free programs. Ferraris [43] also cover more extensive instances of answer set programs, such as arbitrary propositional theories. It obtains the reduct $P^{\mathcal{J}}$ for a program P and interpretation $\mathcal{J}$ by replacing all maximal subformulas in P that are unsatisfied in $\mathcal{J}$ by $\bot$. This corresponds to the *TSR* $\models_F$:

Definition 4.5.23 ($\models_F$)**.** *$\models_F$ extends $\models_{GL}$ with: Let $a^{Aggr} = Agg(\{a_1 : cond_1 \ldots, a_n : cond_n\}) * w$, $(\mathcal{I},\mathcal{J}) \models_F a^{Aggr}$ iff $\mathcal{J} \models_2 a^{Aggr}$ and $\mathcal{I} \models_2 Agg(\{a_i : cond_i \in \{a_1 : cond_1, \ldots, a_n : cond_n\} \mid \mathcal{J} \models_2 cond_i\}) * w$.*

Since $\models_F$ and $\models_{FPL}$ coincide for negation-free programs, an analogous discussion of Example 4.5.15 leads to the conclusion that the satisfaction relation $\models_F$ is not lower-regular nor lower-monotonic.

Proposition 4.5.24. *$\models_F$ extends $\models_2$, i.e., $(\mathcal{I},\mathcal{I}) \models_F \psi$ iff $\mathcal{I} \models_2 \psi$.*

Proof. The truth-assignment is truth-functional, therefore, we only need to prove the property for aggregate atoms. We know that $(\mathcal{I},\mathcal{I}) \models_F a^{Aggr}$ iff $\mathcal{I} \models_2 a^{Aggr}$ and $\mathcal{I} \models_2 Agg(\{a_i : cond_i \in \{a_1 : cond_1, \ldots, a_n : cond_n\} \mid \mathcal{I} \models_2 cond_i\}) * w$. It follows that if $\mathcal{I} \models_2 a^{Aggr}$, then $\mathcal{I} \models_2 Agg(\{a_i : cond_i \in \{a_1 : cond_1, \ldots, a_n : cond_n\} \mid \mathcal{I} \models_2 cond_i\}) * w$. This observation leads to $(\mathcal{I},\mathcal{I}) \models_F a^{Aggr}$ iff $\mathcal{I} \models_2 a^{Aggr}$. □

Proposition 4.5.25. *For convex aggregate atoms, $\models_F$ behaves lower-monotonic, i.e., if $\mathcal{I} \subseteq \mathcal{I}'$ and $(\mathcal{I},\mathcal{J}) \models_F \psi$, then $(\mathcal{I}',\mathcal{J}) \models_F \psi$.*

Proof. $\models_F$ extends $\models_{GL}$ and the latter is lower-monotonic. So we only need to consider a ψ that is a convex aggregate atom. If a^{Aggr} is a convex aggregate atom, then $a^{Aggr}_{\mathcal{J}} = Agg(\{a_i : cond_i \in \{a_1 : cond_1, \ldots, a_n : cond_n\} \mid \mathcal{J} \models_2 cond_i\}) * w$ is also convex since the transformation only deletes conditions from the original convex aggregate. This means that if $\mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \mathcal{I}_3$, $\mathcal{I}_1 \models_2 a^{Aggr}_{\mathcal{J}}$, and $\mathcal{I}_3 \models_2 a^{Aggr}_{\mathcal{J}}$, then $\mathcal{I}_2 \models_2 a^{Aggr}_{\mathcal{J}}$. It is easy to see that if $\mathcal{J} \models_2 a^{Aggr}$, then $\mathcal{J} \models_2 a^{Aggr}_{\mathcal{J}}$. Therefore, if $(\mathcal{I},\mathcal{J}) \models_F a^{Aggr}$ and $(\mathcal{I},\mathcal{J}) \leq_p (\mathcal{I}',\mathcal{J})$, then $\mathcal{I}' \models_2 a^{Aggr}_{\mathcal{J}}$ and thus $(\mathcal{I}',\mathcal{J}) \models_F a^{Aggr}$. □

Definition 4.5.26 (Anti-monotonic aggregate atom)**.** *An aggregate atom a^{Aggr} is anti-monotonic iff for all interpretations $\mathcal{I}, \mathcal{J}$, such that $\mathcal{I} \subseteq \mathcal{J}$, it holds that if $\mathcal{J} \models_2 a^{Aggr}$, then $\mathcal{I} \models_2 a^{Aggr}$.*

Proposition 4.5.27. *For anti-monotonic aggregate atoms, $\models_F$ behaves lower-regular and equivalent with $\models_{ult}$.*

Proof. Let $a^{Aggr}_{\mathcal{J}} = Agg(\{a_i : cond_i \in \{a_1 : cond_1, \ldots, a_n : cond_n\} \mid \mathcal{J} \models_2 cond_i\}) * w$.

1. If $(\mathcal{I}, \mathcal{J}) \models_{ult} a^{Aggr}$, then $\mathcal{I}' \models_2 a^{Aggr}$ for every $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{J}$. Thus, $\mathcal{J} \models a^{Aggr}$ and therefore $\mathcal{J} \models a^{Aggr}_{\mathcal{J}}$. Since a^{Aggr} is anti-monotonic and the only transformation between a^{Aggr} and $a^{Aggr}_{\mathcal{J}}$ deletes conditions, $a^{Aggr}_{\mathcal{J}}$ must be anti-monotonic. Hence $\mathcal{I} \models a^{Aggr}_{\mathcal{J}}$ and $(\mathcal{I}, \mathcal{J}) \models_F a^{Aggr}$.
2. If $(\mathcal{I}, \mathcal{J}) \models_F a^{Aggr}$, then $\mathcal{J} \models_2 a^{Aggr}$. From the anti-monotonicity of a^{Aggr} it follows that for all $\mathcal{I}' \subseteq \mathcal{J}, \mathcal{I}' \models_2 a^{Aggr}$. Hence, $(\mathcal{I}, \mathcal{J}) \models_{ult} a^{Aggr}$.
3. An anti-monotonic aggregate atom is convex and $\models_F$ and $\models_{ult}$ coincide for convex aggregate atoms. Therefore, for anti-monotonic aggregate atoms, $\models_F$ behaves lower-regular.

□

4.5.3 Summary

	LPST	*GZ*[4]	*MR*	*FPL*	*F*
Regular	arbitrary	arbitrary	convex	convex	anti-monotonic
PDB	*ult*	*triv*	*ult*	*ult*	*ult*
lfp	arbitrary	arbitrary	arbitrary	convex	convex

Table 4.1: A summary of the different *TSRs*.

Table 4.1 gives an overview. Row 1 indicates for which aggregate atoms the ternary satisfaction relation behaves lower-regular. Row 2 shows which semantics from Pelov et al. [69] coincides with this semantics for the subclass of programs such that the semantics behaves lower-regular. Row 3 indicates for which aggregate atoms the semantics is monotonic in the first component such that the answer sets can be

[4]Note that the *GZ*-semantics is not defined for programs with conditions under default negation within aggregate-atoms and the table should be interpreted accordingly.

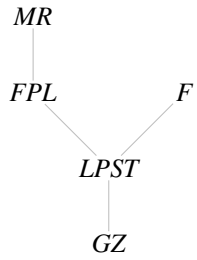


Figure 4.1: A Hasse diagram representing the different *TSRs* ordered by the precision-order, i.e., if a is put below b, then $\models_a \leq_p \models_b$.

constructed as the lfp of Φ_P. Moreover, the Hasse diagram in Section 4.5.3 shows the precision order between the different *TSRs*. If $\models_a \leq_p \models_b$, then every a-answer will always be b-answer set as well. This does not generally hold the other way around. Interestingly, all non-regular semantics coincide with the ult-semantics of Pelov [67] for the subclass of programs in which they behave lower-regular[5]. For programs outside this subclass, the non-regular semantics differ and may derive more answer sets than the ult-semantics.

4.6 Conclusions and future work

Approximation Fixpoint Theory describes various types of constructions from non-monotonic operators and was designed to formalize the view of Logic Programming as constructive definitions, a view at least implicit in stratified logic programs, in the KK and WF semantics, but also in the logic FO(ID) introduced by Denecker and Ternovska [29]. We studied aggregate programs from this view point, showing how regular (three- or four-valued) extensions of the strong Kleene truth assignment induce extensions of KK, WF and stable semantics. We showed that regular truth assignments correspond one-to-one to pairs of a lower-regular and an upper-regular ternary satisfaction relation $(\mathcal{I}, \mathcal{J}) \models_3 \psi$, where the lower-regular one suffices for defining stable models. To study the relation with ASP, we then made a generalized study of *TSRs* as a tool to define answer sets. We analyzed different properties of *TSRs*, and many semantics of aggregate programs in the literature to determine the corresponding lower ternary satisfaction relation and the properties that influence them, such as convexity, (anti-)monotonicity,

[5]Regularity is a property of a semantics. A program cannot be regular or non-regular. However, if a program belongs to a specific subclass of programs, a non-regular semantics may behave lower-regular anyway.

and the sign of conditions in aggregates. We obtained many results linking many ASP semantics in various degrees to the AFT-framework.

In the ASP community, other views of LP and ASP exist than that as a logic of constructive definitions. They are developed in various frameworks such as the framework of **HT** (for more details see the paper by Cabalar et al. [18]) or of Ferraris and Lifschitz (for more details see the paper by Ferraris et al. [44]). These frameworks often extend the original logic programming formalism in various directions, e.g., with disjunction in the head, other negations. They may entirely redefine full FO and the meaning of its connectives. The base idea is that a program corresponds to a theory in some logic (e.g., **HT** or FO) from which answer set are derived using some equilibrium characterization. Although these semantics are not constructive in the sense of AFT, there are surely many interesting mathematical relationships to AFT. For instance, it is striking that the logic of **HT** is also defined in terms of a ternary satisfaction relation. Furthermore, there seemingly exists a correlation between some of the interesting properties of the different *TSRs* and the subset of aggregate-atoms for which they behave lower-regular. For example, we hypothesize that if a *TSR* guarantees $\leq_t$-minimal stable models, it must behave lower-regular for convex aggregate atoms. It is a goal for future work to investigate this.

Chapter 5

Consistent Approximation Fixpoint Theory with Refined Approximation Spaces

5.1 Introduction

As illustrated by the previous chapters, Approximation Fixpoint Theory (AFT) is one of the key mathematical tools for constructive knowledge representation and reasoning. It provides a systematic way to handle partial information and reason about fixpoints by introducing approximations of the exact space. In its standard form (sAFT), AFT uses pairs of elements in the exact space, while the interval-based variant (iAFT) restricts these to consistent pairs, representing lower and upper bounds essential for defining stable and well-founded fixpoints. However, while pairs are the simplest encoding of such information, there is nothing preventing us from considering more refined approximations. By allowing more flexible and refined approximation spaces, we can improve both expressiveness and usability. This chapter develops these ideas and shows how generalized approximation spaces expand the power and applicability of AFT.

The aim of this chapter is twofold. On the one hand, from a practical point of view, the aforementioned approximation spaces do not suffice. Hence, our first goal is to *extend consistent AFT* by allowing a more refined approximation space. On the other hand, once we give room to multiple approximation spaces, we also trigger new questions, i.e., questions regarding the relations between the different types of approximation spaces, or regarding the limits of refining an approximation space. Hence, the second

goal is to *generalize consistent AFT*, offering an overarching framework that allows the investigation of these types of questions.

5.1.1 Extending Consistent AFT

Currently, consistent AFT, i.e., iAFT, represents approximants as intervals $[x, y]$, where x is a lower bound and y an upper bound of a set of exact elements in a complete lattice. This construction has proven both elegant and practical: intervals provide a simple and intuitive way to approximate an element constructed or characterized by the approximator. It approximates each element in it. By a process of improving the approximation, by increasing the lower bound and decreasing the upper bound, the precision increases until an exact point is obtained.

However, despite these strengths, iAFT also has certain limitations. On the one hand, it offers little flexibility with respect to the choice of the exact space itself. By definition, the exact space must be a complete lattice. However, not every natural exact space forms a complete lattice. In fact, in this thesis, we have already established multiple examples of exact spaces that illustrate this, e.g., for the recursive and co-recursive definitions such as the Fibonacci list in Chapter 3. On the other hand, iAFT lacks fine-grainedness: the approximants are restricted to intervals, which means that intermediate or more informative forms of approximation cannot be expressed. In some construction processes, however, the information required to proceed surpasses what can be captured by intervals. These issues motivate us to look beyond intervals towards richer approximation spaces, capable of accommodating finer distinctions and greater flexibility. In this chapter, we develop a generalization of consistent AFT in which the elements of the approximation space need not necessarily be pairs/intervals, but more general approximations are allowed. This additional flexibility allows us to apply more precise and informative approximations. This need is more concretely motivated by some challenging examples from auto-epistemic reasoning, abstract argumentation, more specifically weighted abstract dialectic frameworks [15], and non-monotonic recursive and co-recursive function definitions, which we discuss next.

Inflexibility with Respect to the Exact Space Since intervals are defined relative to a complete lattice of exact elements, iAFT inherits a certain structural rigidity: it forces the construction process to be grounded in a complete lattice, even when the problem domain would naturally suggest another type of poset. However, while many constructions are naturally carried out on a cpo, it is by no means guaranteed that this cpo is also a complete lattice. In fact, this quickly becomes clear by only looking at some examples in this thesis. For example, in the recursive and co-recursive definitions discussed in Chapter 3, the construction space, which serves as the instantiation of the exact space, is a cpo, but not a complete lattice. Notably, iAFT has been successfully

applied to non-monotonic inductive definitions [31], this suggests that AFT might offer key insights in the treatment of (non-monotonic) recursive and co-recursive definitions as well, provided that we can employ it in the first place. Specifically, using AFT can improve our understanding of (co-)recursive definitions by (1) paving the way for non-monotonic semantic operators, and (2) storing information about an upper bound, allowing us to derive exclusion of a possible value during the construction process instead of only when the fixpoint is reached, which may take infinitely many steps.

With this in mind, we now turn to a particularly challenging example: a non-monotonic recursive definition. This example underscores the limitations of requiring the exact space to be a complete lattice and highlights the need for more flexible approximation spaces.

Example 5.1.1. *Let $\mathcal{G} = (V, E)$ be a directed graph with vertices V and edges E. The distance between two vertices x and y is defined as the length of the shortest path between them, if it exists. One can define this distance function $d : V \times V \to \mathbb{N}_\perp$ formally as follows:*

$$\left\lfloor \begin{array}{l} \forall x \in V : \quad d(x,x) := 0. \\ \forall x, z \in V : d(x,z) := \mathit{Min}(S) + 1 \leftarrow S = \left\{ d(x,y) \middle| \begin{array}{l} y \in V : E(y,z) \\ \wedge d(x,y) \neq \perp \end{array} \right\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \wedge x \neq z \wedge S \neq \emptyset. \end{array} \right\rfloor$$

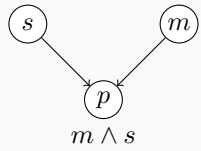
The first rule considers the base case; the shortest path from a node to itself has length 0. The second rule adds one edge to the shortest the path from a vertex x to any vertex y that has an edge to vertex z. Note that, in general, d is partial, as there may be pairs of vertices between which no paths exist.

The construction space for this definition corresponds to all partial functions in ${N_\perp}^{V \times V}$ ordered by the definedness order. The associated semantic operator is not monotonic. In particular, the test of definedness (i.e., $d(x, v) \neq \perp$) in the scope of the negation in the second rule provokes non-monotonicity. The construction space for this definition corresponds to all partial functions in ${\mathbb{N}_\perp}^{V \times V}$ ordered by the definedness order. This is a cpo, but not a complete lattice; there is no 'greatest element'. However, as the semantic operator is non-monotonic, we cannot trust that its monotone inductions will find the expected fixpoint.

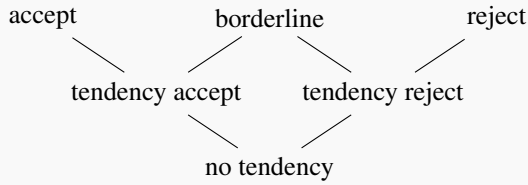
A similar problem also occurs in other domains. To illustrate this, consider weighted abstract dialectical frameworks (wADFs), which were introduced by Brewka et al. [15]. As discussed by Bogaerts [10], iAFT successfully captures the semantics of the covered wADFs. In essence, wADFs are systems that represent knowledge about how different arguments influence each other, i.e., arguments can attack and support one another, possibly by joint attacks or support. In this sense, the acceptance value of one argument

is constructed from the values of its attackers and supporters. An interpretation of a wADF then assigns an *acceptance value* to every argument which represents the degree to which an argument is accepted. For example, if an attacking argument has a high acceptance value, the attacked argument's value decreases. Naturally, this requires an *acceptance order* $\leq_a$ on the set of possible acceptance values. Note the strong resemblance between recursive definitions and wADFs; both construct a function that assigns values to domain elements (arguments) based on its values for other domain elements. However, unlike recursive definitions, where the poset of values has a fixed structure, with a single bottom element representing undefined and multiple maximal elements corresponding to each co-domain value, wADFs allow for more flexible posets, accommodating richer relationships among the values. It is then not surprising that also for wADFs, the exact space can be different from a complete lattice. Due to the fact that sAFT and iAFT use pairs as approximations, Bogaerts [10] had to pose restrictions on the acceptance order. Specifically, the acceptance values equipped with the acceptance order should form a complete lattice. On the other hand, Brewka et al. [15] allows multiple $\leq_a$-maximal acceptance values as illustrated by the following example:

Example 5.1.2 (Based on Example 3.2 as introduced by Brewka et al. [14])**.** *Consider the following wADF, which states that the acceptance status of a paper* (p) *is supported by its significance* (s) *and the used scientific methodology* (m)*. In particular, the value for the status of the paper will be the greatest value that is compatible with the values of the significance and the methodology, i.e., the greatest lower bound of those values* $(m \wedge s)$*.*



(a) A simple wADF illustrating how the status of a paper depends on its significance and the used methodology.



(b) The acceptance values under $\leq_a$.

> *For example, let us assume that the significance of the paper has the value 'accept'. On the other hand, peer reviewing determined that the scientific methodology should be assigned the value 'borderline'. This yields a relatively simple monotonic semantic operator, which maps an assignment of acceptance values for p, s, and m to a new assignment: s is set to 'accept', m to 'borderline', and the value for p is determined based on the current values of s and m. In this case, the intended model assigns the value 'tendency accept' to the paper's status.*
>
> *The order $\leq_a$ can be interpreted as an order on levels of consensus. For example, while both 'borderline' and 'no tendency' are essentially neutral opinions, they are very different in nature. Consider a situation where both reviewers consider a paper 'borderline' in comparison to a situation where one strongly recommends acceptance while the other strongly recommends rejection leading to the value 'no tendency'. The maximal elements are the elements of greatest consensus, i.e., 'accept', 'reject' and 'borderline'.*

In other words, to fully cover all wADFs with AFT we need an approximation space capable of handling exact spaces that are not complete lattices. Alternatively, one can always extend the exact space to a complete lattice. However, such extensions are typically performed in an ad hoc manner, and as we will see, extending exact spaces is far from trivial, and caution is warranted. We will return to this issue in Section 6.3.

Granularity Limitations in iAFT Another important issue arises from the granularity of the approximation space itself. Even when the exact space is suitably structured, intervals can be too coarse to capture the nuances needed in certain construction processes, resulting in imprecise, non-exact well-founded fixpoints. This may happen even if our semantic operator has a very clear intended fixpoint. We now turn our attention to this problem, illustrating it with a concrete example from auto-epistemic logic (AEL).

It has been argued that logic programming under well-founded model semantics can be understood as a logic of inductive definitions [29, 31]. In this, AFT's well-founded fixpoint construction constructs the relations inductively defined by the program. Denecker and Vennekens [31] claim that for such definitional programs, the well-founded model will always identify the defined set. In comparison, in the application of AFT for AEL, the well-founded fixpoint construction does not construct sets, but a belief state. But in some applications, the well-founded fixpoint construction turns out be surprisingly weak and unable to construct the intended object.

Example 5.1.3 (First published by Bogaerts et al. [11], where it is called "Hanne's example", after Hanne Vlaeminck who invented the example)**.** *Consider the following auto-epistemic theory:*

$$\mathcal{T} = \left\{ \begin{array}{l} q \Leftrightarrow \neg K\, p. \\ r \Leftrightarrow \neg K\, q. \end{array} \right\}.$$

This theory is seen as the theory of a rational, epistemic agent with introspection, whose knowledge base does not only contain objective facts of the world, but also propositions about what it knows or does not know. E.g., suppose the theory above would contain the objective fact 'p.' then the agent would know p, i.e., Kp would be true, and according to the agents theory, q would be false, which in turn would lead to r being true according to the theory. But as it happens, the theory does not contain objective information about p, hence the agent does not know p, meaning that Kp is false, and hence q is true, and further down the road, r is false. This is the belief state of the agent: nothing is known about p, but q and $\neg r$ are known to be true. This, at least, is the intended belief state of this theory.

Bogaerts et al. [11] demonstrate that the well-founded semantics based on iAFT fails for this example. We informally revisit the reasons for this failure in Section 5.3. Specifically, there are no suitable strict refinements of the least precise approximant, hence, the well-founded model is this least precise approximant, i.e., the approximant that approximates all considered belief states. Therefore, the authors stated that AFT is not strong enough for some auto-epistemic theories and they proposed an alternative method to determine the semantics of those theories using the concept of well-founded sets. This is a powerful ad-hoc algebraic method that is able to find the expected model of theories of this kind. Intuitively, in that paper, arbitrary sets of elements are used as approximations and a powerful algebraic method is defined that can identify the intuitive model in (a class of theories generalizing) the example above. In this work, on the other hand, we show that the *standard* methods of AFT suffice, provided we generalize them to a richer approximation space.

5.1.2 Generalizing Consistent AFT

Rather than just introducing an additional, more refined approximation space, this chapter identifies necessary conditions for approximation spaces to support AFT without requiring changes in any of the other key-concepts of AFT. In other words, while virtually preserving the standard definitions of key concepts such as 'approximators' and 'the stable operator', we zoom in on the restrictions of the approximation space. As a result, the notion of an approximation space is no longer a fixed construct given an exact space. Instead, our work provides a framework

inducing infinite possibilities for the approximation space. Our new theory is a true generalization of consistent AFT: when instantiated with pairs as approximations, it reduces to the standard definitions of consistent AFT (i.e., iAFT). This generalization provides us with a general overarching framework that allows us to compare, evaluate and better understand different variants of approximation spaces for AFT, offering an interesting and informative new perspective.

5.1.3 Our Contributions

The examples we discussed above demonstrate some of the limitations of AFT with intervals. In order to resolve these limitations, this chapter introduces a more flexible notion of the approximation space enabling the use of more refined approximations. On the one hand, more precise approximations naturally hold more information and therefore, they may result in stronger expressiveness needed to solve, e.g., Example 5.1.3. On the other hand, allowing more flexible approximation spaces allows the use of AFT for logics that assume less restrictive spaces, such as wADFs and (co-)recursive definitions.

The exact spaces in Examples 5.1.1 and 5.1.2 seems to suggest we need approximations with a set of multiple 'maximal elements' as upper bound. Therefore, in Section 5.3, we take a closer look at *flowers* which have more refined upper bounds. We identify a set of interesting and desirable properties of this space which serve as inspiration for the more relaxed definition of an approximation space given in Section 5.4. We show that our extension of AFT is conservative in upholding the core intuitions and theorems of consistent interval-AFT as published by Denecker and co-authors [36, 30], while concurrently expanding the theory's scope by introducing new approximation spaces such as the space of flowers. Using flowers as approximations, AFT is capable of deriving the intended well-founded model for Example 5.1.3. Additionally, flower-based AFT facilitates a straightforward extension of the work by Bogaerts [10] towards any wADFs such as Example 5.1.2. These examples are worked out in detail in Chapter 6. Consequently, the proposed generalization of AFT successfully increases the expressiveness while simultaneously introducing more usability.

On the one hand, the generalization brings about a deeper understanding of (consistent) AFT by zooming in on the necessary conditions. On the other hand, the generalization also introduces a new challenge. Whereas, given a problem in an exact space, the approximation space used to be a fixed entity derivable from the exact space, now an exact space may lead to multiple suitable approximation spaces. This brings up the question of how to select an appropriate approximation space. In Section 5.8, we investigate the relation between approximation spaces within the problem environment. We define a *precision-order* on the approximation spaces such that more precise approximation spaces constitute more accurate semantics provided we use appropriate

approximating operators. However, increased precision comes at a computational cost. Thus, a good approximation space strikes a good balance between computational complexity and expressiveness. In this regard, the results in Section 5.8 showcase that instead of choosing a fixed approximation space from the beginning, we can gradually build a suitable approximation space by iteratively refining the space until it reaches the desired accuracy.

5.2 Preliminaries: AEL and wADFs

In this section we recall the basic notions and definitions of AEL and wADFs to give more context to the motivating Examples 5.1.2 and 5.1.3.

5.2.1 Auto-Epistemic Logic

Auto-epistemic Logic (AEL) is a formalism designed to model the reasoning of an ideally introspective agent, one capable of reflecting on its own beliefs.

Formally, AEL extends propositional logic with a modal operator K, where $K\varphi$ intuitively means 'φ is believed' or 'φ is known'. The language $\mathcal{L}$ in AEL is defined over a finite set of propositional atoms Σ, and its formulas are constructed using the usual Boolean connectives together with the modal operator K.

Definition 5.2.1 (Auto-epistemic theory)**.** *An auto-epistemic theory is a set of formulas in $\mathcal{L}$.*

Just as for standard propositional logics, we will use the term *interpretation* to denote a subset $\mathcal{I} \subseteq \Sigma$. An interpretation represents a possible state of affairs. Typically, some possible states of affairs are consistent with the agent's knowledge while others are not. Those that are consistent with its knowledge form the agent's belief state.

Definition 5.2.2 (Belief state)**.** *The belief state $\mathcal{X}$ of an agent is the set of all interpretations that are believed to be possible according to the agent, i.e., an interpretation $\mathcal{I} \in \mathcal{X}$ is consistent with the knowledge of the agent.*

An AEL theory or formula is evaluated in the context of the belief state of the agent and an interpretation representing the 'real world', i.e., the assumed actual state of affairs.

Definition 5.2.3 (Satisfaction relation)**.** *Let φ be a formula in $\mathcal{L}$, let $\mathcal{X}$ be a belief state and let $\mathcal{I}$ be an interpretation. The satisfaction relation is defined inductively using the standard recursive rules for the satisfaction relation of propositional FO with one additional rule:*

$$\mathcal{X}, \mathcal{I} \models K\varphi \text{ if } \mathcal{X}, \mathcal{J} \models \varphi \text{ for every } \mathcal{J} \in \mathcal{X}.$$

The additional rule formalizes the idea that an agent *knows* a formula if it is true in all the interpretations in its belief state.

Moore [62] defined a semantic operator $O_{\mathcal{T}}$ for each auto-epistemic theory $\mathcal{T}$ mapping a belief state $\mathcal{X}$ to

$$O_{\mathcal{T}}(\mathcal{X}) := \{\mathcal{I} \mid \boldsymbol{X}, \mathcal{I} \models \mathcal{T}\}.$$

This operator forms the base of all work concerning AFT on AEL; it is this semantic operator that is approximated by an approximator. The exact space of interest is the set of belief states ordered by the superset-order.

5.2.2 Weighted Abstract Dialectical Frameworks

Abstract Dialectical Frameworks (ADFs) by Brewka and Woltran [13] generalize Dung's Abstract Argumentation Frameworks (AFs) [40] by allowing arbitrary relations between arguments rather than only attacks. Each node (argument) in an ADF is associated with an acceptance condition or function that specifies how the acceptance of that node depends on its parents.

A Weighted Abstract Dialectical Framework (wADF) introduced by Brewka et al. [15] further extends this idea by allowing degrees of acceptance rather than binary (accepted/rejected) evaluations. Specifically, we assume a *vocabulary* of arguments denoted by S and a poset of acceptance values $\boldsymbol{V}$ that specifies the different degrees of acceptance. Then, an *interpretation* $\mathcal{I}$ of S is a function $S \to \boldsymbol{V}$ that maps all arguments to an acceptance value.

Definition 5.2.4. *Let $\boldsymbol{V} = \langle V, \leq_a \rangle$ be a poset of acceptance values. A weighted Abstract Dialectical Framework is a tuple $\Xi = \langle S, \Phi \rangle$ where:*

- *S is a finite set of arguments,*
- *$\Phi = \{\varphi_s\}_{s \in S}$ is a set of acceptance functions $\varphi_s : (S \to \boldsymbol{V}) \to \boldsymbol{V}$.*

Semantically, a wADF evaluates how consistent or strong the acceptance of each node is, given the acceptance of its parents. This allows nuanced modeling of argument strength and acceptance.

From a theoretical perspective, wADFs can be naturally formalized as operators on lattices of interpretations, making them highly suitable for analysis within AFT as was shown by Bogaerts [10]. Specifically, the acceptance functions Φ together induce a global semantic operator O on the interpretation space V^S which maps an interpretation $\mathcal{I}$ to

$$O(\mathcal{I}) := \{ \ \forall s \in S : s \mapsto \varphi_s(\mathcal{I}).$$

5.3 Approximations with Multiple Maximal Elements

While standard AFT (interval-AFT) already provides sufficient expressiveness for many non-monotonic reasoning logics, Examples 5.1.1 to 5.1.3 showcase the limitations of this theory. In this section we examine a space of approximations sufficiently refined to address the formulated issues for all three examples, i.e., the space of *flowers*. We will refer to this version of AFT as flower-AFT (fAFT). Within this space, we identify and analyze a set of requirements and promising properties which will serve as inspiration for the definition of an approximation space in Section 5.4. Note, however, that while the space serves well as an inspiring and motivating example, it is definitely *not* the only instance of an approximation space covered by our framework.

Let us first consider Example 5.1.3 in a bit more detail. The main semantic objects of AEL are *belief states*; these are sets of interpretations that the agent in question deems possible. The agent *knows* a formula if it is true in all the interpretations in its belief state. For instance, in the belief state that consists of all the interpretations where p is true, Kp holds, as will $K(q \vee \neg q)$, but $K(q \wedge p)$ does not hold. The belief states are ordered by the superset order, hence the greatest belief state $\top$ is the empty set where no interpretation is deemed possible, i.e., the inconsistent belief state. Similarly, the least belief state $\bot$ is the set of all interpretations. Note that 'greatest' and 'least' refer here to the amount of information or knowledge in the belief state. Given a belief state $\mathcal{X}$, we can evaluate all formulas of the form $K\varphi$.

Now it is easy to derive a suitable operator for Example 5.1.3, namely the operator O which maps any belief state $\mathcal{X}$ to the set $\mathcal{Y}$ of all interpretations $\mathcal{I}$ such that *(1)* q is true in $\mathcal{I}$ iff Kp is false in $\mathcal{X}$ and *(2)* r is true in $\mathcal{I}$ iff Kq is false in $\mathcal{X}$. The ultimate approximator in interval-AFT for this operator derives as fixpoints $KK(U) = WF(U) = (\bot, \top)$. Nevertheless, our intuitions for Example 5.1.3 tell us that since there is no direct information about p, p is not known. From this we then derive all other information. In our minds we lower the upper bound for our belief sets, i.e., we only consider belief sets where p is not known. In essence this corresponds to an application refinement. The problem is that, among the remaining belief sets, there are incompatible pairs of belief states, i.e., whose intersection is empty, e.g., any belief state where we know q is incompatible with a belief state where we know $\neg q$. The only approximating pair that approximates all these belief states is $(\bot, \top)$. In other words, intervals are not sufficiently precise to represent the upper bound we had in our mind, therefore we lose all the acquired information. On the other hand, by introducing approximants with an upper bound consisting of multiple maximal elements, we would be able to perfectly represent the set of belief states where p is not known. This leads us to the idea of *flowers*.

Definition 5.3.1 (Lower-bounded Convex Set (Flower))**.** *Given a bounded-complete cpo $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$, the set $S \subseteq \boldsymbol{\mathcal{P}}$ is* lower-bounded convex *if*

- *S is closed and convex, and*
- $\mathrm{glb}(S) \in S$.

We say S is a flower. *We use $\mathcal{F}$ to denote the set of flowers over the bounded-complete cpo $\boldsymbol{\mathcal{P}}$.*

Informally, one can think of flowers as closed unions of sets of intervals that have a common greatest lower bound. We say a flower $\boldsymbol{X}$ approximates an element $x \in \boldsymbol{\mathcal{P}}$ if $x \in \boldsymbol{X}$. Then it is sensible to define a precision-order on $\mathcal{F}$, i.e., given two flowers $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{F}$, then $\boldsymbol{X} \leq_p \boldsymbol{Y}$ if $\boldsymbol{X} \supseteq \boldsymbol{Y}$. It will be shown in Section 5.4 that $\boldsymbol{\mathcal{F}} = \langle \mathcal{F}, \leq_p \rangle$ is a cpo.

The set of belief states where p is not known is a flower. It is a collection of belief states containing at least one world where p is false. This corresponds to the union of all intervals $[\bot, \{w\}]$ where w is an arbitrary world in which p is false.

Suppose we are given a generalized approximator on the set of flowers over $\boldsymbol{\mathcal{C}}$ and we want to extend AFT to work with this. The fact that this set is a cpo is essential if we want to use Theorem 2.2.9 to ensure the Kripke-Kleene fixpoint exists.

Requirement 1. *A generalized approximator must operate on a cpo, i.e., the approximation space must be a cpo.*

A crucial definition in AFT is the stable revision operator. For interval-AFT this operator reasons on a 'lower' and 'upper' element (corresponding to the greatest lower bound and least upper bound of the interval). Hence, in order to stay close to its original definition, we require a decomposition of our approximants into such 'lower' and 'upper' elements.

Requirement 2. *An approximant $\boldsymbol{X}$ must be decomposable into an approximation lower bound (ALB) $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$ and an approximation upper bound (AUB) $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$.*

Naturally a flower $\boldsymbol{X}$ has as an *ALB* $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} = \mathrm{glb}(\boldsymbol{X})$. On the other hand, we can represent the *AUB* $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$ by $Max(\boldsymbol{X})$ since for every $x \in \boldsymbol{X}$ there will exist at least one $m \in Max(\boldsymbol{X})$ such that $x \leq m$. This brings about the decomposition spaces.

Definition 5.3.2 (Decomposition Spaces of Flowers)**.** *Let $\mathcal{F}$ be the set of flowers over a bounded-complete cpo $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$, we define*

- $\mathcal{L}_f = \{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \mid \boldsymbol{X} \in \mathcal{F}\}$.

- $\mathcal{U}_f = \{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \mid \boldsymbol{X} \in \mathcal{F}\}$.

The stable operator is defined as the combination of two *least* fixpoints, one for each decomposition space. This suggests we need an order on our decomposition spaces. Since we think of such elements as bounds of our flowers, it is natural to expect that a more precise flower should correspond to increasing the *ALB* or decreasing the *AUB*. Moreover, since we are extending consistent AFT, we also require that the *ALB* of a flower is smaller than its *AUB*.

Requirement 3. *The (union of the) decomposition spaces should be equipped with an order $\preceq$ such that for every pair of approximants $\boldsymbol{X}$ and $\boldsymbol{Y}$, $\boldsymbol{X} \leq_p \boldsymbol{Y}$ implies $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$.*

Suppose we have two flowers $\boldsymbol{X}$ and $\boldsymbol{Y}$. We know that $\boldsymbol{X} \leq_p \boldsymbol{Y}$ iff $\boldsymbol{X} \supseteq \boldsymbol{Y}$. Clearly, if $\boldsymbol{X} \supseteq \boldsymbol{Y}$, then $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} = \mathrm{glb}(\boldsymbol{X}) \leq \mathrm{glb}(\boldsymbol{Y}) = \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$. Thus, for *ALBs* $l_1, l_2 \in \boldsymbol{\mathcal{L}_f}$ we expect a composition order $\preceq_f$ such that $l_1 \preceq_f l_2$ iff $l_1 \leq l_2$.

The case of *AUBs* of flowers is slightly more complicated. Assuming that $\boldsymbol{X} \supseteq \boldsymbol{Y}$, then for every $m \in Max(\boldsymbol{Y})$, $m \in \boldsymbol{X}$. Consequently, as $\boldsymbol{X}$ is closed, there must exist an $n \in Max(\boldsymbol{X})$ such that $m \leq n$. In other words, $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} = Max(\boldsymbol{Y}) \subseteq Max(\boldsymbol{X})^{\downarrow} = \boldsymbol{X}^{\boldsymbol{\mathcal{U}}\downarrow}$. Thus, for *AUBs* $u_1, u_2 \in \boldsymbol{\mathcal{U}_f}$ we expect a composition order $\preceq_f$ such that $u_1 \preceq_f u_2$ iff $u_1 \subseteq {u_2}^{\downarrow}$.

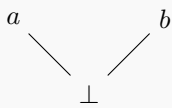
Finally, the composition order should reflect that any *ALB* $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$ of a (consistent) flower is smaller than its *AUB* $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Clearly, for every $l \in \boldsymbol{\mathcal{L}_f}$, there exists a flower $\{l\}$ where $\{l\}^{\boldsymbol{\mathcal{L}}} = l$ and $\{l\}^{\boldsymbol{\mathcal{U}}} = \{l\}$. As a result, $l \preceq_f \{l\}$ must hold. After considering anti-symmetry and transitivity, these observations and expectations are summarized by the following definition.

Definition 5.3.3. *We define the (flower) composition order $\preceq_f$ s.t. for every $b_1, b_2 \in \boldsymbol{\mathcal{L}_f} \cup \boldsymbol{\mathcal{U}_f}$,*

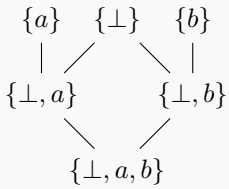
$$b_1 \preceq_f b_2 \text{ if } {b_1}^{\downarrow} \subseteq {b_2}^{\downarrow} \wedge (b_1 \in \boldsymbol{\mathcal{L}_f} \vee b_2 \in \boldsymbol{\mathcal{U}_f}).$$

If for $l_1, l_2 \in \boldsymbol{\mathcal{L}_f}$, $l_1 \preceq_f l_2$, we say l_1 is a less precise *ALB* than l_2. On the other hand, if for $u_1, u_2 \in \boldsymbol{\mathcal{U}_f}$, $u_1 \preceq_f u_2$, we say u_1 is a more precise *AUB* than u_2.

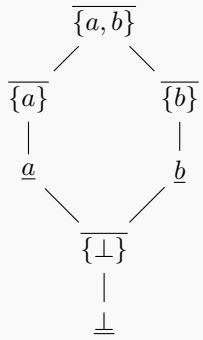
Example 5.3.4. *Suppose we have the following simple bounded-complete cpo.*



For this cpo, the set of non-empty flowers $\mathcal{F}$ consists of $\{\bot\}, \{a\}, \{b\}, \{\bot, a\}, \{\bot, b\}$ and $\{\bot, a, b\}$, as visualized by the following Hasse diagram.



Then $\boldsymbol{\mathcal{L}_f}$ consists of $\bot$, a and b and $\boldsymbol{\mathcal{U}_f}$ consists of $\{\bot\}, \{a\}, \{b\}$ and $\{a, b\}$. Then $\langle \boldsymbol{\mathcal{L}_f} \cup \boldsymbol{\mathcal{U}_f}, \preceq_f \rangle$ is visualized as



where all ALBs are underlined while all AUBs are overlined to visually distinguish between $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$.

Once both fixpoints are determined, the stable revision operator needs to (re)combine the information about the *ALB* and the *AUB* back into a flower. In fact, it is already essential to recombine *ALBs* and *AUBs* to define the operators of which we need the fixpoints in the first place. Clearly, for every $l \in \boldsymbol{\mathcal{L}_f}$ and $u \in \boldsymbol{\mathcal{U}_f}$ such that $l \preceq_f u$, we can recompose them into a flower by taking all elements $x \in \boldsymbol{\mathcal{P}}$ such that $l \leq x$ (i.e., x is greater than the *ALB*) and $x \in u^{\downarrow}$ (i.e., x lies under the *AUB*), i.e., the recomposition $[\![l, u]\!]$ is given by $l^{\uparrow} \cap u^{\downarrow}$. Naturally, this recomposition respects the precision order, i.e., more precise *ALBs* and *AUBs* recompose into more precise flowers.

However, it is important to see that given an arbitrary $l \in \boldsymbol{\mathcal{L}_f}$ and $u \in \boldsymbol{\mathcal{U}_f}$ such that $l \preceq_f u$, decomposing the recomposition $[\![l, u]\!]$ will not always result in the same *ALB* and *AUB*. This is particularly the case whenever there is no flower that

decomposes into both l and u. E.g., consider the cpo in Example 5.3.4, clearly $\{\bot, a, b\} \leq_p \{a\}$, therefore $\bot \preceq_f a \preceq_f \{a\} \preceq_f \{a, b\}$. The recomposition $[\![a, \{a, b\}]\!]$ corresponds to the flower $\{a\}$ since b is not greater than the lower bound a. Therefore, $\{a\}^{\boldsymbol{\mathcal{U}}} = \{a\} \neq \{a, b\}$. Intuitively, recomposing a lower bound and upper bound combines the information on the lower bound and the upper bound to derive additional knowledge about these bounds. It is then natural to expect that such recomposition results in a more precise lower and upper bound after decomposition. In contrast, recomposing the decomposition of a flower $\boldsymbol{X}$ will always obtain the exact same flower, i.e., $\boldsymbol{X} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!] = [\![\mathrm{glb}(\boldsymbol{X}), \mathit{Max}(\boldsymbol{X})]\!]$.

Requirement 4. *We need a recomposition function capable of recombining an ALB $l \in \boldsymbol{\mathcal{L}}$ with an AUB $u \in \boldsymbol{\mathcal{U}}$ into an approximant $[\![l, u]\!]$, provided they are compatible, i.e., $l \preceq u$. Such a recombination might improve the ALB and AUB, but it should never lose information.*

Finally, we need the guaranteed existence of a least fixpoint of the stable revision operator. Moreover, part of the strength of AFT lies in the fact that we can essentially use Markowsky's generalization of the Knaster-Tarski theorem to derive the well-founded model. For this the stable revision operator needs to be a monotonic operator on a cpo. Ideally, this cpo coincides with the approximation space for the least upper bound of its chains, e.g., it contains the least precise approximant. In that case, we do not need to keep track of which approximants are in the subposet.

Requirement 5. *The stable revision operator must be monotonic, well-defined and internal on a cpo which coincides with the approximation space for taking the least upper bounds of its chains.*

Specifically, iAFT identifies the subposet induced by the A-prudent and A-reliable intervals for this. Crucially, the stable operator is well-defined for the A-reliable intervals, which include all fixpoints of the approximator, i.e., the approximants of interest. Moreover, the A-prudent and A-reliable subposet includes all fixpoints of the stable revision operator. As a result, it is assured that the induced cpo contains the least fixpoint of the stable revision operator. Alternatively, we could specifically demand that the stable revision operator is a monotonic and well-defined operator on the cpo induced by the A-reliable and A-prudent approximants. As a direct consequence of the following sections and Appendix A, we find that this alternative requirement warrants the same definition of an approximation space.

5.4 Generalization of Consistent Approximation Spaces

In this section we provide a general notion of an approximation framework and approximation space heavily inspired by the requirements identified in Section 5.3. Later, Sections 5.5 to 5.7 show that the main results of interval-AFT remain valid even after this generalization. Not only are the conditions in the generalized definitions *sufficient* to obtain these results, they are also *necessary* within the context of the aforementioned requirements. In particular, assuming we virtually preserve the original definitions of all other notions relevant to AFT, such as approximators and the stable revision operator, Appendix A shows that every condition posed on the generalized notions is essential to satisfy these requirements.

First, according to Requirement 2, we require that our approximants in the approximation space can be decomposed into an *ALB* and an *AUB*. Moreover, Requirement 3 claims the need for an order on the *ALBs* and *AUBs*. Additionally, although less obvious, to satisfy Requirement 5, the *ALBs* must form a cpo under this order, while the *AUBs* must form a cochain-complete cpo. This is further motivated in Appendix A. We start by formally defining the resulting lower and upper decomposition spaces.

Definition 5.4.1 (Decomposition Spaces)**.** *Let $\mathcal{L}, \mathcal{U}$ be sets and let $\preceq$ be a partial order over $\mathcal{L} \cup \mathcal{U}$. The tuple $\langle \mathcal{L}, \mathcal{U}, \preceq \rangle$ is a* decomposition pair *if*

- *$\langle \mathcal{L} \cup \mathcal{U}, \preceq \rangle$ has a least element $\bot \in \mathcal{L}$,*
- *$\langle \mathcal{L} \cup \mathcal{U}, \preceq \rangle$ has a greatest element $\top \in \mathcal{U}$,*
- *the induced subposet $\boldsymbol{\mathcal{L}} = \langle \mathcal{L}, \preceq \rangle$ is a cpo, and*
- *the induced subposet $\boldsymbol{\mathcal{U}} = \langle \mathcal{U}, \preceq \rangle$ is a cochain-complete cpo.*

We use the terms lower and upper decomposition space *to refer to the posets $\langle \mathcal{L}, \preceq \rangle$ and $\langle \mathcal{U}, \preceq \rangle$, respectively.*

Example 5.4.2. *Let $\mathcal{F}$ be the set of flowers over a bounded-complete cpo $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$. Assume the following definitions as discussed in Section 5.3:*

$$\mathcal{L}_f = \{\mathrm{glb}(\boldsymbol{X}) \mid \boldsymbol{X} \in \mathcal{F}\}$$

$$\mathcal{U}_f = \{Max(\boldsymbol{X}) \mid \boldsymbol{X} \in \mathcal{F}\}$$

$$\forall b_1, b_2 \in \mathcal{L}_f \cup \mathcal{U}_f : b_1 \preceq_f b_2 \text{ if } {b_1}^{\downarrow} \subseteq {b_2}^{\downarrow} \wedge (b_1 \in \mathcal{L}_f \vee b_2 \in \mathcal{U}_f) \quad .$$

The following propositions show that $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$ are indeed decomposition spaces.

Proposition 5.4.3. *$\preceq_f$ is a partial order on $\mathcal{L}_f \cup \mathcal{U}_f$.*

Proof. To improve the readability of this proof we will assume that an element $l \in \boldsymbol{\mathcal{L}_f}$ is represented by the singleton $\{l\}$. We can do that without interfering with the truth of the proposition because of the constraint that $b_1 \in \mathcal{L}_f$ or $b_2 \in \mathcal{U}_f$ in the definition of $\preceq_f$.

Reflexivity and transitivity of $\preceq_f$ follow from the reflexivity and transitivity of $\subseteq$. To show anti-symmetry of $\preceq_f$, we need to show that if ${b_1}^{\downarrow} = {b_2}^{\downarrow}$, then $b_1 = b_2$. We know that $b_1 \subseteq {b_1}^{\downarrow} = {b_2}^{\downarrow} \supseteq b_2$. Thus for every $x_2 \in b_2$ there exists an $x_1 \in b_1$ such that $x_2 \leq x_1$. Similarly, for this $x_1 \in b_1$ there must exist an $x_2' \in b_2$ such that $x_1 \leq x_2'$. Then it follows, $x_2 \leq x_1 \leq x_2'$. Either $b_2 \in \mathcal{L}_f$ and b_2 is a singleton, then $x_2 = x_2'$, or $b_2 \in \mathcal{U}_f$ and b_2 is an anti-chain, therefore it must be the case that if $x_2 \leq x_2'$, then also $x_2 = x_2'$. Thus, in any case $x_2 = x_1 = x_2'$ and therefore $x_2 \in b_1$. Then we find that $b_2 \subseteq b_1$. Analogously, we derive that $b_1 \subseteq b_2$. Then by anti-symmetry of $\subseteq$, we have $b_1 = b_2$. □

Proposition 5.4.4. *If $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$ is bounded-complete, then the poset $\langle \mathcal{L}_f \cup \mathcal{U}_f, \preceq \rangle$ has a least element $\bot = \bot_{\boldsymbol{\mathcal{P}}} \in \boldsymbol{\mathcal{L}_f}$ and a greatest element $\top = Max(\mathcal{P}) \in \boldsymbol{\mathcal{U}_f}$.*

Proof. It is trivial to see that $\bot = \bot_{\boldsymbol{\mathcal{P}}}$ is indeed the least element in this poset, and that $\bot \in \boldsymbol{\mathcal{L}_f}$, by definition of $\preceq_f$. Moreover, by Proposition 2.1.21, we know that since $\boldsymbol{\mathcal{P}}$ is a co-chain-complete cpo, for every subset $S \subseteq \boldsymbol{\mathcal{P}}$, $S^{\downarrow} \subseteq \mathcal{P}^{\downarrow} = Max(\mathcal{P})^{\downarrow}$. Hence, clearly $Max(\mathcal{P})$ is an upper bound of everything in $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$. It is also not hard to see that since $\boldsymbol{\mathcal{P}}$ is a cpo, $\mathcal{P}$ must be flower. Therefore $Max(\mathcal{P}) \in \boldsymbol{\mathcal{U}_f}$. □

Proposition 5.4.5. *If $\boldsymbol{\mathcal{P}}$ is a cpo, then $\boldsymbol{\mathcal{L}_f}$ is a cpo.*

Proof. It is not hard to see that $\boldsymbol{\mathcal{L}_f} = \boldsymbol{\mathcal{P}}$. Clearly, for every element $x \in \boldsymbol{\mathcal{P}}$, $\{x\}$ is a flower, hence $x = \mathrm{glb}(\{x\}) \in \boldsymbol{\mathcal{L}_f}$. On the other hand, for every $x \in \boldsymbol{\mathcal{L}_f}$, there is a non-empty set $\boldsymbol{X}$ (i.e., a flower) such that $x = \mathrm{glb}_{\boldsymbol{\mathcal{P}}}(\boldsymbol{X}) \in \boldsymbol{\mathcal{P}}$. Moreover, it trivially follows from the definition of $.^{\downarrow}$ that for every $x, y \in \boldsymbol{\mathcal{L}_f}$, $x \preceq_f y$ iff $x \leq y$. Therefore, since $\boldsymbol{\mathcal{P}}$ is by assumption a cpo, $\boldsymbol{\mathcal{L}_f}$ must be a cpo as well. □

Lemma 5.4.6. *Let $\boldsymbol{\mathcal{P}}$ be a bounded-complete cpo. If $\boldsymbol{X}$ is a flower over $\boldsymbol{\mathcal{P}}$, then $Max(\boldsymbol{X})^{\downarrow}$ is closed.*

Proof. By definition $Max(\boldsymbol{X})$ is an anti-chain. Take a chain $C \subseteq Max(\boldsymbol{X})^{\downarrow}$. Then it is easy to see that for every $x \in C$ there exists an $m \in Max(\boldsymbol{X})$ such that $x \leq m$. Now, if $\mathrm{glb}(C)$ exists, then $\mathrm{glb}(C) \leq x \leq m$ and therefore $\mathrm{glb}(C) \in Max(\boldsymbol{X})^{\downarrow}$. On the other hand, define the set $C' = \{\mathrm{lub}(\mathrm{glb}(\boldsymbol{X}), x) \mid x \in C\}$. Such a least upper bound must exist since $\boldsymbol{\mathcal{P}}$ is bounded complete and they have a common upper bound. Moreover, it is easy to see that if C is a chain, then C' must be a chain as well. Then for every $y \in C'$, $\mathrm{glb}(\boldsymbol{X}) \leq y$ and $y \in Max(\boldsymbol{X})^{\downarrow}$, thus $y \in \boldsymbol{X}$ since $\boldsymbol{X}$ is convex. Moreover, since $\boldsymbol{X}$ is closed, we know that $\mathrm{lub}(C') \in \boldsymbol{X}$. Then, again since $\boldsymbol{X}$ is closed, there must exist an $m \in Max(\boldsymbol{X})$ such that $\mathrm{lub}(C') \leq m$. Now we find that $\mathrm{lub}(C) \leq \mathrm{lub}(C') \leq m$. Therefore, $\mathrm{lub}(C) \in Max(\boldsymbol{X})$. Consequently, $Max(\boldsymbol{X})^{\downarrow}$ is closed. □

Proposition 5.4.7. *If $\boldsymbol{\mathcal{P}}$ is a cpo, then $\boldsymbol{\mathcal{U}_f}$ is a complete lattice.*

Proof. We will use Proposition 2.1.37 to obtain that $\boldsymbol{\mathcal{U}_f}$ is a complete lattice, in particular, we know by Proposition 5.4.4 that $\boldsymbol{\mathcal{U}_f}$ has a greatest element. Additionally, we will show that every non-empty set has a greatest lower bound.

Given a non-empty set $S \subseteq \boldsymbol{\mathcal{U}_f}$, define

$$\boldsymbol{Y} = \bigcap \{u^{\downarrow} \mid u \in S\}$$

First we will show that $Max(\boldsymbol{Y}) \in \boldsymbol{\mathcal{U}_f}$ by showing that $\boldsymbol{Y}$ is a flower. Since for every $u \in S$, $\perp \in u^{\downarrow}$, $\perp \in \boldsymbol{Y}$. Therefore $\boldsymbol{Y}$ is non-empty and $\mathrm{glb}(\boldsymbol{Y}) = \perp \in \boldsymbol{Y}$. Moreover, since $S \subseteq \boldsymbol{\mathcal{U}_f}$, by Lemma 5.4.6, for every $u \in S$, $u^{\downarrow}$ is closed. By Proposition 2.1.26, the intersection of closed sets is closed as well, thus $\boldsymbol{Y}$ is closed. Similarly, since $\boldsymbol{Y}$ is the intersection of convex sets, by Proposition 2.1.25 $\boldsymbol{Y}$ is convex. Hence, $\boldsymbol{Y}$ is indeed a flower.

Now we will show that $\mathrm{glb}_{\boldsymbol{\mathcal{U}_f}}(S) = Max(\boldsymbol{Y})$. It is not hard to see that since $\boldsymbol{Y}$ is convex and closed, and $\perp \in \boldsymbol{Y}$, $Max(\boldsymbol{Y})^{\downarrow} = \boldsymbol{Y}^{\downarrow} = \boldsymbol{Y}$. Therefore, for every $u \in S$ it follows that $Max(\boldsymbol{Y})^{\downarrow} \subseteq u^{\downarrow}$, thus $Max(\boldsymbol{Y}) \preceq_f u$. Therefore $Max(\boldsymbol{Y})$ is indeed a lower bound of S. Moreover, for every lower bound $u' \in \boldsymbol{\mathcal{U}_f}$ of S, it holds that for every $u \in S$, $u' \preceq_f u$, thus $u'^{\downarrow} \subseteq u^{\downarrow}$, therefore $u'^{\downarrow} \subseteq \bigcap \{u^{\downarrow} \mid u \in S\} = \boldsymbol{Y} = Max(\boldsymbol{Y})^{\downarrow}$ and thus $u' \preceq_f Max(\boldsymbol{Y})$. Hence $\mathrm{glb}_{\boldsymbol{\mathcal{U}_f}}(S) = Max(\boldsymbol{Y})$ □

Corollary 5.4.8. *$\boldsymbol{\mathcal{U}_f}$ is a co-chain-complete cpo.*

In summary, we see that $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$ are a lower and upper decomposition space, respectively.

For the remainder of this chapter, we assume $\boldsymbol{\mathcal{L}} = \langle \mathcal{L}, \preceq \rangle$ and $\boldsymbol{\mathcal{U}} = \langle \mathcal{U}, \preceq \rangle$ are a lower and upper decomposition space. Then we introduce the following definition of a composition poset.

Definition 5.4.9 (Composition poset)**.** *A poset $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$ is a* composition poset *of $\langle \mathcal{L}, \mathcal{U}, \preceq \rangle$, if there exist decomposition-functions $.^{\boldsymbol{\mathcal{L}}} : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{L}}$ and $.^{\boldsymbol{\mathcal{U}}} : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{U}}$, and partial recomposition-function $[\![.,.]\!] : \boldsymbol{\mathcal{L}} \times \boldsymbol{\mathcal{U}} \nrightarrow \boldsymbol{\mathcal{A}}$ such that for every $l, l_1, l_2 \in \boldsymbol{\mathcal{L}}$ and $u, u_1, u_2 \in \boldsymbol{\mathcal{U}}$:*

1. *$[\![l, u]\!]$ is defined iff $l \preceq u$.*
2. *$l \preceq [\![l, u]\!]^{\boldsymbol{\mathcal{L}}}$ and $[\![l, u]\!]^{\boldsymbol{\mathcal{U}}} \preceq u$.*
3. *if $l_1 \preceq l_2 \preceq u_2 \preceq u_1$, then $[\![l_1, u_1]\!] \leq_p [\![l_2, u_2]\!]$.*
4. *for every $\boldsymbol{X}, \boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$, if $\boldsymbol{X} \leq_p \boldsymbol{Y}$, then $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$.*
5. *for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, $\boldsymbol{X} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]$.*
6. *$[\![l, \top]\!]^{\boldsymbol{\mathcal{L}}} = l$ and $[\![\bot, u]\!]^{\boldsymbol{\mathcal{U}}} = u$.*

In natural language, we will refer to $\leq_p$ as the precision order, and to $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$ and $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$ as the ALB and AUB of $\boldsymbol{X}$, respectively.

In this definition, condition 1 follows trivially from Requirement 4. Furthermore, condition 2 entails the second part of this requirement, i.e., a recomposition can only refine the information. Information is never lost by recomposing an *ALB* with an *AUB*. Additionally, conditions 3 to 4 ensure that the precision order corresponds to the natural, expected relation where we consider the *ALB* and *AUB* to respectively act as a type of lower bound and upper bound of the approximant. Essentially, they guarantee the desired (anti-)monotonicity properties of $.^{\boldsymbol{\mathcal{L}}}$, $.^{\boldsymbol{\mathcal{U}}}$ and $[\![.,.]\!]$.

Proposition 5.4.10. *If $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$ is a composition poset of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$, then*

- *the decomposition function $.^{\boldsymbol{\mathcal{L}}} : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{L}}$ is monotonic.*
- *the decomposition function $.^{\boldsymbol{\mathcal{U}}} : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{U}}$ is anti-monotonic.*
- *the recomposition function $[\![.,.]\!] : \boldsymbol{\mathcal{L}} \times \boldsymbol{\mathcal{U}} \nrightarrow \boldsymbol{\mathcal{A}}$ is monotonic in the first argument and anti-monotonic in the second argument.*

where the notion of (anti-)monotonicity of functions is defined analogously as for operators.

Proof. This follows trivially from conditions 3 to 4 in the definition of a composition poset. □

Moreover, condition 5 secures that the recomposition and decomposition functions are correct abstractions of our intuitions. In particular, the *ALB* $\boldsymbol{X}^{\mathcal{L}}$ and *AUB* $\boldsymbol{X}^{\mathcal{U}}$ should completely characterize the approximant $\boldsymbol{X}$. Therefore, combined, they are expected to hold the same information as $\boldsymbol{X}$, i.e., $[\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!] = \boldsymbol{X}$. Finally, condition 6 is essential to procure that $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ correspond to *ALBs* and *AUBs* of approximants in our approximation space, as expected. For example, assume for some $l \in \boldsymbol{\mathcal{L}}$ that $[\![l, \top]\!]^{\mathcal{L}} \neq l$. By condition 3, for any $u \in \boldsymbol{\mathcal{U}}$ such that $l \preceq u$, $[\![l, \top]\!] \leq_p [\![l, u]\!]$. Therefore, by conditions 2 and 4, $l \preceq [\![l, \top]\!]^{\mathcal{L}} \preceq [\![l, u]\!]^{\mathcal{L}}$. Consequently, since $l \neq [\![l, \top]\!]^{\mathcal{L}}$, also $l \neq [\![l, u]\!]^{\mathcal{L}}$. In other words, there is no approximant $\boldsymbol{X}$ such that $\boldsymbol{X}^{\mathcal{L}} = l$, otherwise, by condition 5, $l = \boldsymbol{X}^{\mathcal{L}} = [\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!]^{\mathcal{L}} = [\![l, \boldsymbol{X}^{\mathcal{U}}]\!]^{\mathcal{L}}$. In summary, l is not an *ALB* of any approximant $\boldsymbol{X}$. A similar reasoning holds for $\boldsymbol{\mathcal{U}}$.

Example 5.4.11 (Example 5.4.2, contd.)**.** *Assume the following definitions of the decomposition and recomposition functions for fAFT as discussed in Section 5.3:*

$$\forall \boldsymbol{X} \in \mathcal{F} : \boldsymbol{X}^{\mathcal{L}} = \mathrm{glb}(\boldsymbol{X})$$

$$\forall \boldsymbol{X} \in \mathcal{F} : \boldsymbol{X}^{\mathcal{U}} = Max(\boldsymbol{X})$$

$$\forall l \in \mathcal{L}_f, u \in \mathcal{U}_f : [\![l, u]\!] = l^{\uparrow} \cap u^{\downarrow} \text{ if } l \preceq_f u$$

Condition 1 in Definition 5.4.23 follows trivially from the definition of $[\![.,.]\!]$.

Proposition 5.4.12. *If* $l \in \boldsymbol{\mathcal{L}_f}$ *and* $u \in \boldsymbol{\mathcal{U}_f}$ *such that* $l \preceq_f u$*, then* $[\![l, u]\!]^{\mathcal{L}} = \mathrm{glb}_{\boldsymbol{\mathcal{P}}}([\![l, u]\!]) = l$.

Proof. Take $x \in [\![l, u]\!]$, then by definition of $[\![.,.]\!]$, $x \in l^{\uparrow}$, thus $l \leq x$. In other words l is a lower bound of $[\![l, u]\!]$. Moreover, since $l \preceq_f u$, $l^{\downarrow} \subseteq u^{\downarrow}$, hence $l \in u^{\downarrow}$. As a result, $l \in u^{\downarrow} \cap l^{\uparrow} = [\![l, u]\!]$. In summary, l must be the greatest lower bound of $[\![l, u]\!]$. □

Proposition 5.4.13. *If* $l \in \boldsymbol{\mathcal{L}_f}$ *and* $u \in \boldsymbol{\mathcal{U}_f}$*, then* $[\![l, u]\!]^{\mathcal{U}} = Max([\![l, u]\!]) = u \cap l^{\uparrow}$.

Proof. We start by proving that $Max([\![l, u]\!]) \subseteq u \cap l^{\uparrow}$, afterwards we show the other direction. We have $Max([\![l, u]\!]) \subseteq [\![l, u]\!] = l^{\uparrow} \cap u^{\downarrow}$. Therefore we only have to show that $Max([\![l, u]\!]) \subseteq u$. Let $m \in Max([\![l, u]\!])$. It follows that $m \in u^{\downarrow}$ and $l \leq m$. Therefore, there must exist an $u \in U$ such that $l \leq m \leq u$. Then $u \in l^{\uparrow}$, thus $u \in [\![l, u]\!]$. Since m was a maximal element of $[\![l, u]\!]$, this

entails that $u = m$ and thus $m \in u$. Therefore $Max([\![l, u]\!]) \subseteq u$ and thus $Max([\![l, u]\!]) = Max([\![l, u]\!]) \cap u \subseteq l^{\uparrow} \cap u^{\downarrow} \cap u = l^{\uparrow} \cap u$.

Now we will show that also $u \cap l^{\uparrow} \subseteq Max([\![l, u]\!])$. Clearly, $u \cap l^{\uparrow} \subseteq [\![l, u]\!]$. Take $x \in u \cap l^{\uparrow}$. Let us assume that x is not a maximal element of $[\![l, u]\!]$, i.e., there exists an $y \in [\![l, u]\!]$ such that $x < y$. Since $y \in [\![l, u]\!]$, also $y \in u^{\downarrow}$. Then there must exist an $z \in u$ such that $y \leq z$. Therefore $x < y \leq z$. This would mean that u is not an anti-chain, and therefore $u \notin \boldsymbol{\mathcal{U}_f}$. Therefore x must be a maximal element. Since x was arbitrarily chosen, this implies $u \cap l^{\uparrow} \subseteq Max([\![l, u]\!])$. □

Corollary 5.4.14. *If $l \in \boldsymbol{\mathcal{L}_f}$ and $u \in \boldsymbol{\mathcal{U}_f}$, then $[\![l, u]\!]^{\boldsymbol{\mathcal{U}}} \preceq_f u$.*

Proof. This trivially follows from the fact that $[\![l, u]\!]^{\boldsymbol{\mathcal{U}}} \subseteq u$, by Proposition 5.4.13. □

As a result, we can conclude that the definitions above also satisfy condition 2. By definition of $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$, it is clear that for every flower $\boldsymbol{X}$, $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \in \boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \in \boldsymbol{\mathcal{U}_f}$. Additionally, let us show that when defined, $[\![., .]\!]$ indeed maps its arguments to a flower.

Proposition 5.4.15. *For every $l \in \boldsymbol{\mathcal{L}_f}$ and $u \in \boldsymbol{\mathcal{U}_f}$, $[\![l, u]\!] \in \mathcal{F}$.*

Proof. Clearly since $l \preceq_f u$, $l \in [\![l, u]\!]$, thus $[\![l, u]\!]$ is not empty. Moreover, by Lemma 5.4.6 $u^{\downarrow}$ is closed, it is also easy to see that $l^{\uparrow}$ is closed since $\boldsymbol{\mathcal{P}}$ is a bounded-complete cpo. By Proposition 2.1.26 finite intersection of closed sets is closed as well, therefore $[\![l, u]\!]$ is closed. Similarly, by Proposition 2.1.25 the intersection of convex sets is convex as well. Finally, by Proposition 5.4.12, we have that $\mathrm{glb}([\![l, u]\!]) = l \in [\![l, u]\!]$. Then it follows that $[\![l, u]\!]$ is a flower. □

Proposition 5.4.16. *Let $l_1, l_2 \in \boldsymbol{\mathcal{L}_f}$ and let $u_1, u_2 \in \boldsymbol{\mathcal{U}_f}$. If $l_1 \preceq_f l_2 \preceq_f u_2 \preceq_f u_1$, then $[\![l_1, u_1]\!] \supseteq [\![l_2, u_2]\!]$.*

Proof. We have already shown that $l_1 \preceq_f l_2$ iff $l_1 \leq l_2$. Hence, ${l_1}^{\uparrow} \supseteq {l_2}^{\uparrow}$. On the other hand, $u_2 \preceq_f u_1$ implies by definition that ${u_2}^{\downarrow} \subseteq {u_1}^{\downarrow}$. Therefore, it is trivial to see that $[\![l_1, u_1]\!] = {l_1}^{\uparrow} \cap {u_1}^{\downarrow} \supseteq {l_2}^{\uparrow} \cap {u_2}^{\downarrow} = [\![l_2, u_2]\!]$. □

The previous proposition then trivially implies conditions 3 in the definition of a composition poset. Moreover, the de- and recomposition functions also satisfy condition 5 as claimed by the following proposition.

Proposition 5.4.17. *For every $\boldsymbol{X} \in \mathcal{F}$, $\boldsymbol{X} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]$.*

Proof. Let $x \in \boldsymbol{X}$. Then $x \in \boldsymbol{X}^{\downarrow} = Max(\boldsymbol{X})^{\downarrow} = \boldsymbol{X}^{\boldsymbol{\mathcal{U}}\downarrow}$. Moreover, $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} = \mathrm{glb}(\boldsymbol{X}) \leq x$, hence $x \in \boldsymbol{X}^{\boldsymbol{\mathcal{L}}\uparrow}$. Consequently, $x \in \boldsymbol{X}^{\boldsymbol{\mathcal{U}}\downarrow} \cap \boldsymbol{X}^{\boldsymbol{\mathcal{L}}\uparrow} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]$. The other direction follows straightforwardly from the fact that $\boldsymbol{X}$ is convex, and $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \in \boldsymbol{X}$ and $Max(\boldsymbol{X}) \subseteq \boldsymbol{X}$. Therefore $\boldsymbol{X} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]$. □

Proposition 5.4.18. *Let $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{F}$. If $\boldsymbol{X} \supseteq \boldsymbol{Y}$, then $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq_f \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq_f \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq_f \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$.*

Proof. Since $\boldsymbol{X} \supseteq \boldsymbol{Y}$ and $\boldsymbol{Y}$ is non-empty, it trivially follows that $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} = \mathrm{glb}_{\boldsymbol{\mathcal{P}}}(\boldsymbol{X}) \leq \mathrm{glb}_{\boldsymbol{\mathcal{P}}}(\boldsymbol{Y}) = \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$. Moreover, since $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} = Max(\boldsymbol{Y}) \subseteq \boldsymbol{Y}$, it is easy to see that $\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} = \mathrm{glb}_{\boldsymbol{\mathcal{P}}}(\boldsymbol{Y}) \in Max(\boldsymbol{Y})^{\downarrow} = \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}\downarrow}$. Therefore $\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq_f \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$. Let $y \in \boldsymbol{Y}^{\downarrow}$, then there exists an $x \in \boldsymbol{Y}$ such that $y \leq x$. Since $\boldsymbol{Y} \subseteq \boldsymbol{X}$, $x \in \boldsymbol{X}$. Consequently, $y \in \boldsymbol{X}^{\downarrow}$. Since we arbitrarily chose y, this holds for every $y \in \boldsymbol{Y}^{\downarrow}$. Therefore $\boldsymbol{Y}^{\downarrow} \subseteq \boldsymbol{X}^{\downarrow}$. As a result, by Proposition 2.1.21, $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}\downarrow} = Max(\boldsymbol{Y})^{\downarrow} = \boldsymbol{Y}^{\downarrow} \subseteq \boldsymbol{X}^{\downarrow} = Max(\boldsymbol{X})^{\downarrow} = \boldsymbol{X}^{\boldsymbol{\mathcal{U}}\downarrow}$. Hence, $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq_f \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. □

Proposition 5.4.19. *If $l \in \boldsymbol{\mathcal{L}_f}$ and $u \in \boldsymbol{\mathcal{U}_f}$, then $[\![l, Max(\mathcal{P})]\!]^{\boldsymbol{\mathcal{L}}} = l$ and $[\![\bot, u]\!]^{\boldsymbol{\mathcal{U}}} = u$.*

Proof. Clearly, by Proposition 5.4.4, $l \preceq_f Max(\mathcal{P})$, hence, by Proposition 5.4.12, $[\![l, Max(\mathcal{P})]\!]^{\boldsymbol{\mathcal{L}}} = l$. Similarly, by Proposition 5.4.4, $\bot \preceq_f u$. Therefore, by Proposition 5.4.13, $[\![\bot, u]\!]^{\boldsymbol{\mathcal{U}}} = u \cap \bot^{\uparrow} = u \cap \mathcal{P} = u$. □

Consequently, as expected $\langle \mathcal{F}, \supseteq \rangle$ is a composition poset of $\langle \mathcal{L}_f, \mathcal{U}_f, \preceq_f \rangle$.

The composition poset of flowers is particularly interesting because it can gain information by combining a lower bound (*ALB*) with an upper bound (*AUB*). Specifically, the combination might further refine the upper bound as explained in Section 5.3. However, it is important to note that refining such upper bound essentially exposes previously implicit knowledge. As such, we cannot refine our upper bound beyond the point where all knowledge is explicit. This is formalized by condition 5 in the definition of a composition poset. Interestingly, this condition then implies that every combination of bounds conveying a level of explicit information between that of the original bounds and the refined bounds, will result in the same approximant in the end, as formalized by the following proposition.

Proposition 5.4.20. *Let $\boldsymbol{\mathcal{A}}$ be a composition poset of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ and let $l, l' \in \boldsymbol{\mathcal{L}}$ and $u, u' \in \boldsymbol{\mathcal{U}}$. If $l \preceq l' \preceq [\![l, u]\!]^{\boldsymbol{\mathcal{L}}} \preceq [\![l, u]\!]^{\boldsymbol{\mathcal{U}}} \preceq u' \preceq u$, then*

$$[\![l', u']\!] = [\![l, u]\!].$$

Proof. By conditions 3, $[\![l,u]\!] \leq_p [\![l',u']\!] \leq_p [\![[\![l,u]\!]^{\mathcal{L}}, [\![l,u]\!]^{\mathcal{U}}]\!] = [\![l,u]\!]$ where the last equality follows from condition 5. □

Example 5.4.21. *In the case of iAFT on a complete lattice $\mathcal{P} = \langle \mathcal{P}, \leq \rangle$, we have the following:*

$\mathcal{L} = \mathcal{P} = \mathcal{U}$

$\mathcal{A} = \mathcal{P}^c = \{(l,u) \in \mathcal{P}^2 \mid l \leq u\}$

$\forall (l_1,u_1),(l_2,u_2) \in \mathcal{P}^c : (l_1,u_1) \leq_p (l_2,u_2)$ *iff* $l_1 \leq l_2 \wedge u_2 \leq u_1$

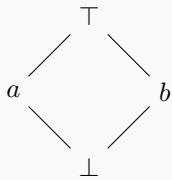
$\forall (l,u) \in \mathcal{P}^c : [\![l,u]\!] = (l,u)$

$\forall (l,u) \in \mathcal{P}^c : (l,u)^{\mathcal{L}} = (l,u)^1 = l$

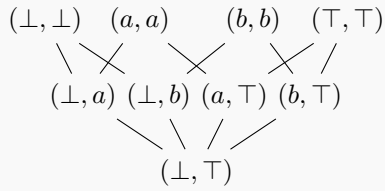
$\forall (l,u) \in \mathcal{P}^c : (l,u)^{\mathcal{U}} = (l,u)^2 = u$

Then it is easy to see that $\langle \mathcal{P}^c, \leq_p \rangle$ is indeed a composition poset of $\mathcal{P}$ with itself. In particular, since $\langle \mathcal{P}, \leq \rangle$ is a complete lattice, it is also a (cochain-complete) cpo. Moreover, it has a least and greatest element. Furthermore, by definition $\mathcal{P}^c = \{(l,u) \in \mathcal{P}^2 \mid l \leq u\}$, hence $[\![l,u]\!]$ is defined iff $l \leq u$. All other conditions follow straightforwardly from the definitions of the re- and decomposition functions.

For example, suppose $\mathcal{P}$ corresponds to the following simple complete lattice.



This is also the poset $\langle \mathcal{L} \cup \mathcal{U}, \preceq \rangle$. The composition poset $\langle \mathcal{P}^c, \leq_p \rangle$ is visualized as:



Example 5.4.22. *As a final example, we look at the simplest composition poset (up to isomorphism). Consider the following definitions:*

$\mathcal{L} = \{\bot\}$ *and* $\mathcal{U} = \{\top\}$

$\preceq = \{(\bot, \bot), (\top, \top), (\bot, \top)\}$

We can visualize the resulting simple poset $\langle \boldsymbol{\mathcal{L}} \cup \boldsymbol{\mathcal{U}}, \preceq \rangle$ *as follows*



The composition poset is even simpler and only consists of a single element.

$\mathcal{A} = \{[\![\bot, \top]\!]\}$

$\leq_p = \{([\![\bot, \top]\!], [\![\bot, \top]\!])\}$

$[\![\bot, \top]\!]^{\boldsymbol{\mathcal{L}}} = \bot$ *and* $[\![\bot, \top]\!]^{\boldsymbol{\mathcal{U}}} = \top$

It is trivial to see that in the context of these definitions, $\boldsymbol{\mathcal{A}}$ *is indeed a composition poset of* $\boldsymbol{\mathcal{L}}$ *and* $\boldsymbol{\mathcal{U}}$*.*

With the construct of a composition poset, we can define the notion of an approximation framework, explicitly identifying when a composition poset warrants a well-defined, monotonic stable operator given a monotonic approximator.

Definition 5.4.23 (Approximation framework)**.** *An* approximation framework $\mathcal{A}$ *is a tuple* $\langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ *such that*

1. $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$ *is a composition poset of* $\langle \boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{U}}, \preceq \rangle$*.*
2. *it satisfies the* Chain Interlattice Lub Property*: Let* $u \in \boldsymbol{\mathcal{U}}$ *and let* $C \subseteq \boldsymbol{\mathcal{L}}$ *be a chain. If for every* $l \in C$*,* $l \preceq u$*, then* $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \preceq u$*.*

3. *it satisfies the* Interlattice Lb Property*: Let $l \in \boldsymbol{\mathcal{L}}$ and let $S \subseteq \boldsymbol{\mathcal{U}}$. If for every $u \in S$, $l \preceq u$, then there exists an $u' \in \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(S)$ such that $l \preceq u'$.*

Note the strong resemblance between, respectively, the Chain Interlattice Lub Property and the Interlattice Lb Property, and the Interlattice Lub Property and Interlattice Glb Property as proposed by Charalambidis et al. [22]. In fact, if $\boldsymbol{\mathcal{U}}$ is a complete lattice, then the Interlattice Lb Property coincides with the Interlattice Glb Property.

Proposition 5.4.24. *The Interlattice Lub Property implies the Weak Interlattice Glb Property, i.e., let $S \subseteq \boldsymbol{\mathcal{U}}$ and $l \in \boldsymbol{\mathcal{L}}$. If $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S)$ exists and for every $u \in S$, $l \leq u$, then $l \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S)$.*

Proof. Since for every $u \in S$, $l \preceq u$, the Interlattice Lub Property implies there exists an $u' \in \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(S)$ such that $l \preceq u'$. By definition of the greatest lower bound, $u' \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S)$. This entails $l \preceq u' \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S)$. □

Informally, the Chain Interlattice Lub Property and Interlattice Lb Property ensure that refining the *ALB* or *AUB* preserves the structure of our restricted decomposition spaces. In particular, when we restrict our original cpo $\boldsymbol{\mathcal{L}}$ to everything below an *AUB* $u \in \boldsymbol{\mathcal{U}}$, it remains a cpo. Similarly, the Interlattice Lb Property guarantees that the restriction of the bounded-complete cpo $\boldsymbol{\mathcal{U}}$ to everything above an *ALB* $l \in \boldsymbol{\mathcal{L}}$ again results in a co-chain complete cpo.

Given an approximation framework, it is trivial to define the space of approximants, i.e., the approximation space.

Definition 5.4.25 (Approximation Space and Approximants)**.** *Let $\mathbb{A} = \langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ be an approximation framework. We call $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$ the* approximation space*, and any $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ an* approximant*.*

In the remainder of the chapter, we assume that $\mathbb{A} = \langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ is an approximation framework with approximation space $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$.

Example 5.4.26 (Example 5.4.21, contd.)**.** *Given a complete lattice $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$, the tuple $\langle \mathcal{P}, \mathcal{P}, \mathcal{P}^c, \leq, \leq_p \rangle$ is an approximation framework with approximation space $\boldsymbol{\mathcal{P}^c} = \langle \mathcal{P}^c, \leq_p \rangle$. The Chain Interlattice Lub Property and Interlattice Lb Property follow straightforwardly from the observation that $\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}}$. We will use the notation $\boldsymbol{\mathcal{A}_i}(\boldsymbol{\mathcal{P}})$ to denote the approximation space for $\boldsymbol{\mathcal{P}}$ in a context of iAFT, i.e., $\boldsymbol{\mathcal{A}_i}(\boldsymbol{\mathcal{P}}) = \boldsymbol{\mathcal{P}^c} = \langle \mathcal{P}^c, \leq_p \rangle$.*

Example 5.4.27 (Example 5.4.22, contd.)**.** *Similarly, given any poset $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \leq \rangle$, the tuple $\langle \{\bot\}, \{\top\}, \{[\![\bot, \top]\!]\}, \preceq, \leq_p \rangle$ is an approximation framework with a singleton as approximation space. The Chain Interlattice Lub Property and Interlattice Lb Property follow trivially from the fact that $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ are singletons. We will refer to this approximation space as the trivial approximation space and we will denote it by $\boldsymbol{\mathcal{A}}_{triv}$.*

Example 5.4.28 (Example 5.4.2, contd.)**.** *The tuple $\langle \mathcal{L}_f, \mathcal{U}_f, \mathcal{F}, \preceq_f, \supseteq \rangle$ is an approximation framework with approximation space $\boldsymbol{\mathcal{F}} = \langle \mathcal{F}, \supseteq \rangle$. We already claimed that $\boldsymbol{\mathcal{F}}$ is a composition poset of $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$. Moreover, we will also show that the Chain Interlattice Lub Property and the Interlattice Lb Property are satisfied. Note, however, that the analogous version of this last property for upper bounds does not necessarily hold. For example, consider the bounded-complete cpo in Example 5.3.4 where we add a greatest element $\top$ to obtain a complete lattice. Then $a \preceq \{a, b\}$ and $b \preceq_f \{a, b\}$, however, $\mathrm{lub}_{\boldsymbol{\mathcal{L}_f}}\{a, b\} = \top \not\preceq \{a, b\}$.*

Proposition 5.4.29. *The tuple $\langle \boldsymbol{\mathcal{L}_f}, \boldsymbol{\mathcal{U}_f}, \mathcal{F}, \preceq_f, \supseteq \rangle$ satisfies the Chain Interlattice Lub Property.*

Proof. Let $u \in \boldsymbol{\mathcal{U}_f}$ and let $C \subseteq \boldsymbol{\mathcal{L}_f}$ be a chain such that for every $l \in C$, $l \preceq_f u$. Then $l \in u^{\downarrow}$. Therefore C is a chain in $u^{\downarrow} \subseteq \boldsymbol{\mathcal{P}}$. Since $\boldsymbol{\mathcal{P}}$ is a cpo, this entails $\mathrm{lub}_{\boldsymbol{\mathcal{P}}}(C)$ exists. Moreover, since $u^{\downarrow}$ is closed, $\mathrm{lub}_{\boldsymbol{\mathcal{P}}}(C) \in u^{\downarrow}$. Hence $\mathrm{lub}_{\boldsymbol{\mathcal{L}_f}}(C) = \mathrm{lub}_{\boldsymbol{\mathcal{P}}}(C) \preceq_f u$. □

Proposition 5.4.30. *The tuple $\langle \boldsymbol{\mathcal{L}_f}, \boldsymbol{\mathcal{U}_f}, \mathcal{F}, \preceq_f, \supseteq \rangle$ satisfies the Interlattice Lb Property.*

Proof. Let $l \in \boldsymbol{\mathcal{L}_f}$ and let $S \subseteq \boldsymbol{\mathcal{U}_f}$ such that for every $u \in S$, $l \preceq_f u$. From the proof of Proposition 5.4.7, we know that $\mathrm{glb}_{\boldsymbol{\mathcal{U}_f}}(S) = Max(\boldsymbol{Y})$ with $\boldsymbol{Y} = \bigcap\{u^{\downarrow} \mid u \in S\}$. Take an arbitrary $u \in S$, since $l \preceq_f u$, we have $l \in l^{\downarrow} \subseteq u^{\downarrow}$. Since u was chosen arbitrarily, we know this is the case for all elements in S, therefore $l \in \boldsymbol{Y}$. Since $\boldsymbol{Y}$ is an intersection of closed sets, $\boldsymbol{Y}$ must be a closed set. Therefore there must exist an $m \in Max(\boldsymbol{Y})$ such that $l \leq m$. Then it is trivial to see that $l^{\downarrow} \subseteq Max(\boldsymbol{Y})^{\downarrow}$, therefore $l \preceq_f Max(\boldsymbol{Y}) = \mathrm{glb}_{\boldsymbol{\mathcal{U}_f}}(S)$. □

We will use the notation $\boldsymbol{\mathcal{A}_f}(\boldsymbol{\mathcal{P}})$ to denote the approximation space for $\boldsymbol{\mathcal{P}}$ in a context of fAFT, i.e., $\boldsymbol{\mathcal{A}_f}(\boldsymbol{\mathcal{P}}) = \boldsymbol{\mathcal{F}} = \langle \mathcal{F}, \supseteq \rangle$.

Additionally, the idea of a composition poset and the decomposition functions allows us to define a generalized notion of the truth order that stays close to the original definition. It is not difficult to see that this relation is indeed an order.

Definition 5.4.31 (Truth Order)**.** *The* truth order *is a binary relation $\leq_t$ on the approximation space where for every $\boldsymbol{X}, \boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$, $\boldsymbol{X} \leq_t \boldsymbol{Y}$ iff $\boldsymbol{X}^{\mathcal{L}} \preceq \boldsymbol{Y}^{\mathcal{L}}$ and $\boldsymbol{X}^{\mathcal{U}} \preceq \boldsymbol{Y}^{\mathcal{U}}$.*

Our goal was to build a conservative extension of iAFT. Therefore, the remaining principal definitions in this chapter are natural generalizations of their counterparts in interval-AFT. As such, from this point forward, the challenge lies in proving that these generalizations are well-defined and that the main theorems still hold.

We start by showing that the approximation space $\boldsymbol{\mathcal{A}}$ is a cpo, as desired. On the one hand, we will prove that $\boldsymbol{\mathcal{A}}$ has a least element $\bot_{\boldsymbol{\mathcal{A}}}$. Additionally, every non-empty chain in $\boldsymbol{\mathcal{A}}$ has a least upper bound.

Proposition 5.4.32. *The approximation space $\boldsymbol{\mathcal{A}}$ has a least element $\bot_{\boldsymbol{\mathcal{A}}} = [\![\bot, \top]\!]$.*

Proof. We know that for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, $\bot \preceq \boldsymbol{X}^{\mathcal{L}} \preceq \boldsymbol{X}^{\mathcal{U}} \preceq \top$. Therefore by the (anti-)monotonicity properties of the recomposition function, $[\![\bot, \top]\!] \leq_p [\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!] = \boldsymbol{X}$. This is the case for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, thus, $[\![\bot, \top]\!]$ must be the least element $\bot_{\boldsymbol{\mathcal{A}}}$ of $\boldsymbol{\mathcal{A}}$. □

Now, we will shift our focus to non-empty chains in $\boldsymbol{\mathcal{A}}$. For this, we make explicit some interesting relations between the approximation space and the decomposition spaces. In particular, we are interested in the existence of the least upper bound and greatest lower bound of a set of approximants in relation to those of the sets of point-wise decompositions. To simplify notation, we first lift the decomposition functions to sets.

Definition 5.4.33. *For each subset $S \subseteq \boldsymbol{\mathcal{A}}$, we define*

$$S^{\mathcal{L}} = \{\boldsymbol{X}^{\mathcal{L}} \mid \boldsymbol{X} \in S\}$$

$$S^{\mathcal{U}} = \{\boldsymbol{X}^{\mathcal{U}} \mid \boldsymbol{X} \in S\}$$

Lemma 5.4.34. *Assume $S \subseteq \boldsymbol{\mathcal{A}}$.*

- *If $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}})$ and $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})$ exist, and $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}}) \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})$, then $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(S)$ exists and $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(S) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}}), \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})]\!]$.*
- *If $\mathrm{glb}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}})$ and $\mathrm{lub}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})$ exist, and S is non-empty, then $\mathrm{glb}_{\boldsymbol{\mathcal{A}}}(S)$ exists and $\mathrm{glb}_{\boldsymbol{\mathcal{A}}}(S) = [\![\mathrm{glb}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}}), \mathrm{lub}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})]\!]$.*

Proof. We first prove the first statement. Let us assume $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}})$ and $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})$ exist, and $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}}) \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})$. Note that by condition 1 in the definition of a composition poset $[\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(S^{\mathcal{L}}), \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S^{\mathcal{U}})]\!]$ is well-defined. We first prove

that $[\![\mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}})]\!]$ is an upper bound of S. Then we show that for any upper bound $\boldsymbol{Y}$ of S, $[\![\mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}})]\!] \leq_p \boldsymbol{Y}$. Then it follows that $[\![\mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}})]\!]$ must be the least upper bound of S.

- We know that for an arbitrary $\boldsymbol{X} \in S$, $\boldsymbol{X}^{\mathcal{L}} \in S^{\mathcal{L}}$, thus $\boldsymbol{X}^{\mathcal{L}} \preceq \mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}})$. Analogously, we find $\mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}}) \preceq \boldsymbol{X}^{\mathcal{U}}$. Thus, by the (anti-)monotonicity properties of $[\![.,.]\!]$,

  $$\boldsymbol{X} = [\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!] \leq_p [\![\mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}})]\!].$$

  This holds for every $\boldsymbol{X} \in S$, thus $[\![\mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}})]\!]$ is an upper bound of S.

- Since $\boldsymbol{Y}$ is an upper bound of S, for every $\boldsymbol{X} \in S$, $\boldsymbol{X} \leq_p \boldsymbol{Y}$, hence by monotonicity of $.^{\mathcal{L}}$ and anti-monotonicity of $.^{\mathcal{U}}$, $\boldsymbol{X}^{\mathcal{L}} \preceq \boldsymbol{Y}^{\mathcal{L}}$ and $\boldsymbol{Y}^{\mathcal{U}} \preceq \boldsymbol{X}^{\mathcal{U}}$. Since this is the case for any $\boldsymbol{X} \in S$, this entails that $\boldsymbol{Y}^{\mathcal{L}}$ is an upper bound of $S^{\mathcal{L}}$ and $\boldsymbol{Y}^{\mathcal{U}}$ is a lower bound of $S^{\mathcal{U}}$. Therefore,

  $$\mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}}) \preceq \boldsymbol{Y}^{\mathcal{L}} \text{ and } \boldsymbol{Y}^{\mathcal{U}} \preceq \mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}}).$$

  Now it trivially follows from the (anti-)monotonicity properties of $[\![.,.]\!]$ that $[\![\mathrm{lub}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{glb}_{\mathcal{U}}(S^{\mathcal{U}})]\!] \leq_p [\![\boldsymbol{Y}^{\mathcal{L}}, \boldsymbol{Y}^{\mathcal{U}}]\!] = \boldsymbol{Y}$.

The proof of the second statement is similar. Assume $\mathrm{glb}_{\mathcal{L}}(S^{\mathcal{L}})$ and $\mathrm{lub}_{\mathcal{U}}(S^{\mathcal{U}})$ exist. Take any $\boldsymbol{X} \in S$ (which is possible as S is non-empty) such that $\mathrm{glb}_{\mathcal{L}}(S^{\mathcal{L}}) \preceq \boldsymbol{X}^{\mathcal{L}} \preceq \boldsymbol{X}^{\mathcal{U}} \preceq \mathrm{lub}_{\mathcal{U}}(S^{\mathcal{U}})$. Hence, $[\![\mathrm{glb}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{lub}_{\mathcal{U}}(S^{\mathcal{U}})]\!]$ is well-defined. Everything else is completely analogous. First we can prove that $[\![\mathrm{glb}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{lub}_{\mathcal{U}}(S^{\mathcal{U}})]\!]$ is a lower bound of S. We can also show that for any lower bound $\boldsymbol{Y}$ of S, $\boldsymbol{Y} \leq_p [\![\mathrm{glb}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{lub}_{\mathcal{U}}(S^{\mathcal{U}})]\!]$. Then it follows that $[\![\mathrm{glb}_{\mathcal{L}}(S^{\mathcal{L}}), \mathrm{lub}_{\mathcal{U}}(S^{\mathcal{U}})]\!]$ must be the greatest lower bound of S. □

Building further upon Lemma 5.4.34, we can finally show that every non-empty chain of approximants must indeed have a least upper bound.

Proposition 5.4.35. *If $C \subseteq \boldsymbol{\mathcal{A}}$ is a chain, then $C^{\mathcal{L}}$ and $C^{\mathcal{U}}$ are chains.*

Proof. This follows straightforwardly from the monotonicity and anti-monotonicity of $.^{\mathcal{L}}$ and $.^{\mathcal{U}}$, respectively. □

Proposition 5.4.36. *If $C \subseteq \boldsymbol{\mathcal{A}}$ is a chain, then $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C)$ exists and $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C) = [\![\mathrm{lub}_{\mathcal{L}}(C^{\mathcal{L}}), \mathrm{glb}_{\mathcal{U}}(C^{\mathcal{U}})]\!]$.*

Proof. We need to prove that $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}})$ and $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})$ exist. Moreover, we need to show that $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}) \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})$. Then by Lemma 5.4.34, it straightforwardly follows that $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C)$ exists and $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})]\!]$.

Since $\boldsymbol{\mathcal{L}}$ is a cpo and $C^{\boldsymbol{\mathcal{L}}}$ is a chain, $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}})$ exists. Similarly, since $\boldsymbol{\mathcal{U}}$ is a co-cpo and $C^{\boldsymbol{\mathcal{U}}}$ is a chain, $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})$ exists.

Take an arbitrary $\boldsymbol{X} \in C$. Since C is a non-empty chain, for every $\boldsymbol{Y} \in C$ either $\boldsymbol{X} \leq_p \boldsymbol{Y}$ or $\boldsymbol{Y} \leq_p \boldsymbol{X}$. In the first case $\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$, while in the second case $\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Consequently, for every $\boldsymbol{Y} \in C$,

$$\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}},$$

hence $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$ is an upper bound of $C^{\boldsymbol{\mathcal{L}}}$. Therefore, by the Chain Interlattice Lub Property,

$$\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}.$$

Since this holds for arbitrary $\boldsymbol{X} \in C$ and $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})$ exists, by the Weak Interlattice Glb Property,

$$\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}) \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}}).$$

Hence, $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C)$ exists and $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})]\!]$. □

Proposition 5.4.37. *The approximation space $\langle \boldsymbol{\mathcal{A}}, \leq_p \rangle$ is a cpo.*

Proof. This trivially follows from Proposition 5.4.32 and Proposition 5.4.36. □

> **Example 5.4.38** (5.4.2, 5.4.21 and 5.4.22, contd.)**.** *In summary, just like the poset of intervals (ordered by the super set order) for iAFT, the trivial approximation space and the poset of flowers, i.e., $\langle \mathcal{F}, \supseteq \rangle$ are cpos. Hence, Markowsky's theorem is applicable.*

Theorem 2.2.9 subsequently entails that every monotonic operator on the approximation space has a least fixpoint. Hence, if, analogously to iAFT, the approximator is $\leq_p$-monotonic, then Theorem 2.2.9 guarantees the existence of the Kripke-Kleene fixpoint.

Definition 5.4.39 (Approximating Operator (Approximator))**.** *We say that $A : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{A}}$ is an* approximating operator *or* approximator *if A is $\leq_p$-monotonic. We use $Appr(\boldsymbol{\mathcal{A}})$ to denote the set of approximators for $\boldsymbol{\mathcal{A}}$.*

The discussion of the generalized results regarding the stable and well-founded fixpoints is broken down into three parts. In Section 5.5, we follow the more algebraic

approach and decidedly prove all relevant propositions and theorems concerning a generalized stable revision operator. Then, Section 5.6 discusses how the generalized notions of an approximation space, approximator and stable revision operator translate to practical applications of non-monotonic semantic operators on an exact space. Finally, Section 5.7 focuses on well-founded inductions, simultaneously providing a constructive way to find the well-founded fixpoint and an informal semantics to generalized AFT.

5.5 Generalization of the Stable Revision Operator

The stable revision operator is a key ingredient in AFT. On the one hand, it is used to ensure the existence of a well-founded fixpoint and stable fixpoint. On the other hand, it provides us with a computationally appealing algorithm to calculate such fixpoint in practice, i.e., by using a monotonic induction of the stable revision operator. Specifically, both of these stem from the observation that the stable revision operator is a well-defined monotonic operator on a cpo. Therefore, it is crucial we prove this for the generalization of the stable revision operator as well. We start by conservatively generalizing the stable revision operator. The remainder of the section essentially proves that this generalization satisfies Requirement 5.

In Section 5.5.1 we show that the generalized stable revision operator is well-defined on the subposet of A-reliable approximants. Afterwards, in Section 5.5.2, we additionally show this operator is monotonic on the subposet of A-reliable and A-prudent pairs. Finally, Section 5.5.3 discusses how the latter subposet is a cpo which coincides with the approximation space for its least upper bounds, and which is closed under the stable revision operator. In summary, we can conclude Markowsky's generalization of the Knaster-Tarski theorem is applicable to the stable revision operator on the space of A-reliable and A-prudent approximants, mirroring the results for iAFT.

Let us first introduce some auxiliary concepts. In particular, traditionally, the stable operator was defined using two auxiliary functions, one on (a subset of) the 'lower bounds', i.e., $\boldsymbol{\mathcal{L}}$, and the other on (a subset of) the 'upper bounds', i.e., $\boldsymbol{\mathcal{U}}$.

Definition 5.5.1. *Given $a, b \in \boldsymbol{\mathcal{L}} \cup \boldsymbol{\mathcal{U}}$, we define*

$$[a, b]_{\boldsymbol{\mathcal{L}}} = \{l \in \boldsymbol{\mathcal{L}} \mid a \preceq l \preceq b\}$$

$$[a, b]_{\boldsymbol{\mathcal{U}}} = \{u \in \boldsymbol{\mathcal{U}} \mid a \preceq u \preceq b\}$$

Definition 5.5.2. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$ and let $\boldsymbol{X}, \boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$. The projector functions $A^{\boldsymbol{\mathcal{L}}} : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{L}}$ and $A^{\boldsymbol{\mathcal{U}}} : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{U}}$ are defined such that if $A(\boldsymbol{X}) = \boldsymbol{Y}$, then $A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}) = \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$ and $A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}) = \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$.*

Definition 5.5.3. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$ and let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ be an approximant. We define the functions*

- $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}} : [\bot, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]_{\boldsymbol{\mathcal{L}}} \to \boldsymbol{\mathcal{L}} : l \mapsto A^{\boldsymbol{\mathcal{L}}}([\![l, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!])$.
- $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}} : [\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \top]_{\boldsymbol{\mathcal{U}}} \to \boldsymbol{\mathcal{U}} : u \mapsto A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, u]\!])$.

The aforementioned auxiliary functions allow us to conservatively extend the notion of the stable revision operator. Note again that we virtually preserved this definition from previous renditions.

Definition 5.5.4 (Stable Revision Operator)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. The* stable revision operator $St_A : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{A}}$ *maps any approximant $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ to $[\![St^1_A(\boldsymbol{X}), St^2_A(\boldsymbol{X})]\!]$, where*

$$St^1_A(\boldsymbol{X}) = \mathrm{lfp}(A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}})$$

$$St^2_A(\boldsymbol{X}) = \mathrm{lfp}(A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}})$$

In the remainder of the chapter we will drop the subscript A when the approximating operator is clear from the context.

5.5.1 Well-defined Stable Revision Operator

To ensure the stable revision operator is well-defined for any A-reliable approximant $\boldsymbol{X}$, we need to establish that the least fixpoints of $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$ and $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$ exist. For this, we will again invoke Markowsky's generalization of the Knaster-Tarski theorem in Theorem 2.2.9. First, it is not hard to see that $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$ and $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$ are monotonic.

Proposition 5.5.5. *If $A \in Appr(\boldsymbol{\mathcal{A}})$ and $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, then $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$ and $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$ are monotonic functions.*

Proof. This trivially follows from the monotonicity of A, and the (anti-)monotonicity properties of the recomposition and decomposition functions. □

Moreover, it is not difficult to show that the sub-posets induced by $[\bot, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]_{\boldsymbol{\mathcal{L}}}$ and $[\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \top]_{\boldsymbol{\mathcal{U}}}$ are cpos. In fact, as argued in our informal justification to include the Chain Interlattice Lub Property and the Interlattice Lb Property, we notice that refining the *ALB* or *AUB* is essentially structure-preserving as clarified by the following propositions.

Proposition 5.5.6. *For every $l \in \boldsymbol{\mathcal{L}}$ and $u \in \boldsymbol{\mathcal{U}}$ such that $l \preceq u$, $\langle [l, u]_{\boldsymbol{\mathcal{L}}}, \preceq \rangle$ is a cpo where for every non-empty chain C, $\mathrm{lub}_{[l,u]_{\boldsymbol{\mathcal{L}}}}(C) = \mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C)$.*

Proof. Clearly for every $l' \in [l,u]_{\boldsymbol{\mathcal{L}}}$, $l \preceq l'$. Moreover $l \preceq l \preceq u$, thus $l \in [l,u]_{\boldsymbol{\mathcal{L}}}$, therefore $[l,u]_{\boldsymbol{\mathcal{L}}}$ has a least element l.

Let C be a non-empty chain in $[l,u]_{\boldsymbol{\mathcal{L}}} \subseteq \boldsymbol{\mathcal{L}}$. Since $\boldsymbol{\mathcal{L}}$ is a cpo, there exists a $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \in \boldsymbol{\mathcal{L}}$. Since $C \subseteq [l,u]_{\boldsymbol{\mathcal{L}}}$, clearly for every $l' \in C$, $l \preceq l' \preceq u$. By the Interlattice Chain Lub Property, this implies $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \preceq u$. Thus $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \in [l,u]_{\boldsymbol{\mathcal{L}}}$. Therefore, $\langle [l,u]_{\boldsymbol{\mathcal{L}}}, \preceq \rangle$ is a cpo. □

Corollary 5.5.7. *For every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, $\langle [\bot, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]_{\boldsymbol{\mathcal{L}}}, \preceq \rangle$ is a cpo where for every chain C, $\mathrm{lub}_{[\bot, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]_{\boldsymbol{\mathcal{L}}}}(C) = \mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C)$.*

Proposition 5.5.8. *The Interlattice Lb Property implies that for every $l \in \boldsymbol{\mathcal{L}}$: $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}([l,\top]_{\boldsymbol{\mathcal{U}}})$ exists and $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}([l,\top]_{\boldsymbol{\mathcal{U}}})$ is the least element of $[l,\top]_{\boldsymbol{\mathcal{U}}}$.*

Proof. Take $S = [l,\top]_{\boldsymbol{\mathcal{U}}}$. Then for every $u \in S$, $l \preceq u$ by choice of S. Hence, by the Interlattice Lb Property, there exists an $u' \in \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(S)$ such that $l \preceq u'$. Clearly, by choice of S, $u' \in S$, therefore for every $b \in \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(S)$, $b \preceq u'$, hence u' is the greatest lower bound of S in $\boldsymbol{\mathcal{U}}$. Thus $u' = \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(S) \in S$. Consequently, u' must be the least element in $S = [l,\top]_{\boldsymbol{\mathcal{U}}}$. □

Proposition 5.5.9. *For every $l \in \boldsymbol{\mathcal{L}}$, $\langle [l,\top]_{\boldsymbol{\mathcal{U}}}, \preceq \rangle$ is a cpo where for every non-empty chain C, $\mathrm{lub}_{[l,\top]_{\boldsymbol{\mathcal{U}}}}(C) = \mathrm{lub}_{\boldsymbol{\mathcal{U}}}(C)$.*

Proof. From Proposition 5.5.8, it follows that $[l,\top]_{\boldsymbol{\mathcal{U}}}$ has a least element, i.e., $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}([l,\top]_{\boldsymbol{\mathcal{U}}})$. Let us assume $C \subseteq [l,\top]_{\boldsymbol{\mathcal{U}}}$ is a non-empty chain. Since C is a chain in $\boldsymbol{\mathcal{U}}$, $\mathrm{lub}_{\boldsymbol{\mathcal{U}}}(C)$ exists. Moreover, since C is not empty, there exists an $u \in C$, such that $l \preceq u \preceq \mathrm{lub}_{\boldsymbol{\mathcal{U}}}(C)$, hence $\mathrm{lub}_{\boldsymbol{\mathcal{U}}}(C) \in [l,\top]_{\boldsymbol{\mathcal{U}}}$. □

Proposition 5.5.10. *For every $l \in \boldsymbol{\mathcal{L}}$, $\langle [l,\top]_{\boldsymbol{\mathcal{U}}}, \preceq \rangle$ is a co-cpo where for every chain C, $\mathrm{glb}_{[l,\top]_{\boldsymbol{\mathcal{U}}}}(C) = \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$.*

Proof. Clearly $l \preceq \top \preceq \top$ and $\top \in \boldsymbol{\mathcal{U}}$, thus $\top \in [l,\top]_{\boldsymbol{\mathcal{U}}}$, therefore $[l,\top]_{\boldsymbol{\mathcal{U}}}$ has a greatest element $\top$. Let C be a non-empty chain of $[l,\top]_{\boldsymbol{\mathcal{U}}} \subseteq \boldsymbol{\mathcal{U}}$. Since $\boldsymbol{\mathcal{U}}$ is a co-cpo, there exists a $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C) \in \boldsymbol{\mathcal{U}}$. Since $C \subseteq [l,\top]_{\boldsymbol{\mathcal{U}}}$, clearly for every $u \in C$, $l \preceq u$. By the Interlatice Glb Property, this implies $l \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$. Thus $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C) \in [l,\top]_{\boldsymbol{\mathcal{U}}}$. □

In contrast, it is considerably less obvious that these auxiliary functions are *operators* on a cpo. In particular, it is entirely possible that $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$ maps an $l \in [\bot, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]_{\boldsymbol{\mathcal{L}}}$ to some element outside of its domain, i.e., $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(l) \notin [\bot, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]_{\boldsymbol{\mathcal{L}}}$. Similarly, $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$ is not necessarily internal on its domain $[\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \top]_{\boldsymbol{\mathcal{U}}}$. For iAFT it suffices to impose A-reliability of $\boldsymbol{X}$ to obtain internal operators. This continues to be sufficient to guarantee that $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$ is internal on $[\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \top]_{\boldsymbol{\mathcal{U}}}$.

Definition 5.5.11 (A-Reliable)**.** *Let A be an approximating operator. An approximant $\boldsymbol{X}$ is A-reliable if $\boldsymbol{X} \leq_p A(\boldsymbol{X})$.*

Proposition 5.5.12. *Let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ be an approximant. If $\boldsymbol{X}$ is A-reliable, then for every $u \in [\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}}$,*

$$\boldsymbol{X}^{\mathcal{L}} \preceq A^{\mathcal{U}}([\![\boldsymbol{X}^{\mathcal{L}}, u]\!]) \preceq \top,$$

i.e., $A^{\mathcal{U}}([\![\boldsymbol{X}^{\mathcal{L}}, u]\!]) \in [\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}}$, hence $A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$ is internal on $[\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}}$.

Proof. From Proposition 5.5.8, we know that $u' = \mathrm{glb}_{\mathcal{U}}([\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}})$ exists. Clearly, since $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, $\boldsymbol{X}^{\mathcal{L}} \preceq \boldsymbol{X}^{\mathcal{U}} \preceq \top$, thus $\boldsymbol{X}^{\mathcal{U}} \in [\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}}$, thus $u' \preceq \boldsymbol{X}^{\mathcal{U}}$. By the Weak Interlattice Glb Property, we also get that $\boldsymbol{X}^{\mathcal{L}} \preceq u'$. Moreover, since $u \in [\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}}$, $u' \preceq u$. It now follows:

$$\begin{aligned}
\boldsymbol{X}^{\mathcal{L}} \preceq A^{\mathcal{L}}(\boldsymbol{X}) = A^{\mathcal{L}}([\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!]) && (\boldsymbol{X} \text{ is } A\text{-reliable}) \\
\preceq A^{\mathcal{L}}([\![\boldsymbol{X}^{\mathcal{L}}, u']\!]) && (A \text{ is monotonic and } u' \preceq \boldsymbol{X}^{\mathcal{U}}) \\
\preceq A^{\mathcal{U}}([\![\boldsymbol{X}^{\mathcal{L}}, u']\!]) && (A([\![\boldsymbol{X}^{\mathcal{L}}, u']\!]) \in \boldsymbol{\mathcal{A}}) \\
\preceq A^{\mathcal{U}}([\![\boldsymbol{X}^{\mathcal{L}}, u]\!]) && (A \text{ is monotonic and } u' \preceq u)
\end{aligned}$$

□

In comparison, A-reliability of $\boldsymbol{X}$ no longer fully addresses the analogous concern for $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$. However, it still is adequate to ensure that $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ is internal on $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$, entailing that $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ must have a fixpoint in $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$.

Lemma 5.5.13. *Let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$. If $\boldsymbol{X}$ is A-reliable, then for every $l \in [\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$,*

$$\boldsymbol{X}^{\mathcal{L}} \preceq A^{\mathcal{L}}([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \preceq \boldsymbol{X}^{\mathcal{U}},$$

i.e., $A^{\mathcal{L}}([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \in [\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$, hence $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ is internal on $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$.

Proof. Since $l \in [\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$, $\boldsymbol{X}^{\mathcal{L}} \preceq l \preceq \boldsymbol{X}^{\mathcal{U}}$. It now follows:

$$\begin{aligned}
\boldsymbol{X}^{\mathcal{L}} \preceq A^{\mathcal{L}}(\boldsymbol{X}) = A^{\mathcal{L}}([\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!]) && (\boldsymbol{X} \text{ is } A\text{-reliable}) \\
\preceq A^{\mathcal{L}}([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) && (A \text{ is monotonic and } \boldsymbol{X}^{\mathcal{L}} \preceq l) \\
\preceq A^{\mathcal{U}}([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) && (A([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \in \boldsymbol{\mathcal{A}}) \\
\preceq A^{\mathcal{U}}([\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!]) && (A \text{ is monotonic and } \boldsymbol{X}^{\mathcal{L}} \preceq l)
\end{aligned}$$

$$\preceq \boldsymbol{X}^{\mathcal{U}} \qquad (\boldsymbol{X} \text{ is } A\text{-reliable})$$

□

Lemma 5.5.14. *Let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$. If $\boldsymbol{X}$ is A-reliable, then the restriction of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ to $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$ has a least fixpoint.*

Proof. By Proposition 5.5.6 $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$ is a cpo. Moreover, by Lemma 5.5.13, $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ is internal on $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$. Then, by Theorem 2.2.9, the restriction of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ to $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$ must have a least fixpoint. □

Notably, this result entails that $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ must have a fixpoint l^*. This in turn ensures the existence of a least fixpoint. Essentially, we can exploit the fixpoint l^* to identify another subset $[\bot, l^*]_{\mathcal{L}} \subseteq [\bot, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$ for which $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ is internal, i.e., the *$\boldsymbol{X}$-complying ALBs*. Critically, this set induces a cpo such that the least upper bound of a chain coincides with its least upper bound in $\boldsymbol{\mathcal{L}}$. Note that this additional notion of $\boldsymbol{X}$-compliance merely serves as a tool in proofs, it has no bearing on the actual theory behind AFT.

Definition 5.5.15. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$ and let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ be A-reliable. Let l^* be the least fixpoint of the restriction of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ to $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$. An ALB $l \in \boldsymbol{\mathcal{L}}$ is $\boldsymbol{X}$-*complying *for A if $l \preceq l^*$. We use the notation $\boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$ to denote the $\boldsymbol{X}$-complying subset of $\boldsymbol{\mathcal{L}}$.*

Lemma 5.5.16. *Given an approximant $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ and approximator A, then $\langle \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A, \preceq \rangle$ is a cpo where for every chain $C \subseteq \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$, $\mathrm{lub}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A}(C) = \mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C)$.*

Proof. Let l^* be the least fixpoint of the restriction of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ to $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$. Clearly, since $\bot$ is the least element, $\bot \preceq l^*$. Hence, $\bot \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$. Therefore $\langle \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A, \preceq \rangle$ has a least element $\bot$.

Now we only need to prove that for every non-empty chain $C \subseteq \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$, $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$. This is trivial. Note that C is also a chain in $\boldsymbol{\mathcal{L}}$, therefore $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C)$ is well-defined. Moreover, for every $c \in C$, $c \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$, therefore $c \preceq l^*$. Hence, l^* is an upper bound of C. By definition of the least upper bound, this entails $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \preceq l^*$. Consequently, $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$. □

Lemma 5.5.17. *Let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ be an approximant. If $\boldsymbol{X}$ is A-reliable, then for every $l \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$, $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}(l) = A^{\mathcal{L}}([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$.*

Proof. Let l^* be the least fixpoint of the restriction of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ to $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$. Since $l \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$, $l \preceq l^* \preceq \boldsymbol{X}^{\mathcal{U}}$. It now follows:

$$A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}(l) = A^{\mathcal{L}}([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \preceq A^{\mathcal{L}}([\![l^*, \boldsymbol{X}^{\mathcal{U}}]\!]) \qquad (A \text{ is monotonic and } l \preceq l^*)$$

$$= A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}(l^*) = l^* \qquad (l^* \text{ is a fixpoint of } A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}})$$

□

At this point we have obtained all prerequisites to deploy Markowsky's theorem, guaranteeing that $St^1(\boldsymbol{X})$ and $St^2(\boldsymbol{X})$ are well-defined as long as the approximant $\boldsymbol{X}$ is A-reliable.

Proposition 5.5.18. *If an approximant $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-reliable, then $St^1(\boldsymbol{X})$ is well-defined.*

Proof. By Lemma 5.5.17 and Proposition 5.5.5, we know that $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ is a monotonic operator internal on $\boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$. Moreover, by Lemma 5.5.16 we know that $\langle \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A, \preceq \rangle$ is a cpo. Therefore by Theorem 2.2.9, $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ must have a least fixpoint in $\langle \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A, \preceq \rangle$. Since for every chain $C \in \boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$, $\mathrm{lub}_{\mathcal{L}}(C) = \mathrm{lub}_{\boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A}(C)$, by Proposition 2.2.13 this least fixpoint must correspond to the least fixpoint of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ in $\langle [\bot, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}, \preceq \rangle$. Therefore, $St^1(\boldsymbol{X})$ is well-defined. □

Proposition 5.5.19. *If an approximant $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-reliable, then $St^2(\boldsymbol{X})$ is well-defined.*

Proof. By Proposition 5.5.12 and Proposition 5.5.5, we know that $A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$ is a monotonic operator internal on $[\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}}$. Moreover, by Proposition 5.5.9, we know that $[\boldsymbol{X}^{\mathcal{L}}, \top]_{\mathcal{U}}$ is a cpo. Therefore by Theorem 2.2.9, $A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$ must have a least fixpoint, hence $\mathrm{lfp}(A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}})$ exists and $St^2(\boldsymbol{X})$ is well-defined. □

Finally, all that remains to be established is the consistency of the stable revision operator for A-reliable approximants $\boldsymbol{X}$, i.e., for $St(\boldsymbol{X}) = [\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!]$ to be well-defined, we require $St^1(\boldsymbol{X}) \preceq St^2(\boldsymbol{X})$. For this, we first introduce the following lemma.

Lemma 5.5.20. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$ and let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$. If $\boldsymbol{X}$ is A-reliable, then $St^1(\boldsymbol{X}) \preceq \boldsymbol{X}^{\mathcal{U}}$, $\boldsymbol{X}^{\mathcal{L}} \preceq St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\mathcal{U}}$.*

Proof. Let l^* be the least fixpoint of the restriction of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ to $[\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$. Clearly $l^* \preceq \boldsymbol{X}^{\mathcal{U}}$. From Proposition 2.2.13, it follows that St^1 is equal to the least fixpoint of the restriction of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ to $\boldsymbol{\mathcal{L}}^{\boldsymbol{X}}_A$. Therefore, $St^1(\boldsymbol{X}) \preceq l^* \preceq \boldsymbol{X}^{\mathcal{U}}$.

Additionally, by definition of St^2 and Proposition 5.5.12, $\boldsymbol{X}^{\mathcal{L}} \preceq St^2(\boldsymbol{X})$. Moreover, by A-reliability of $\boldsymbol{X}$ we have $\boldsymbol{X} \leq_p A(\boldsymbol{X})$, thus $A^{\mathcal{U}}(\boldsymbol{X}) \preceq \boldsymbol{X}^{\mathcal{U}}$. Thus, $\boldsymbol{X}^{\mathcal{U}}$ is a pre-fixpoint of $A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$. Since $St^2(\boldsymbol{X})$ is the least pre-fixpoint of $A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$, we find that $St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\mathcal{U}}$. □

Proposition 5.5.21. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$ and let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$. If $\boldsymbol{X}$ is A-reliable, then $St^1(\boldsymbol{X}) \preceq St^2(\boldsymbol{X})$ and $St(\boldsymbol{X}) \in \boldsymbol{\mathcal{A}}$.*

Proof. We only need to show that $St^1(\boldsymbol{X}) \preceq St^2(\boldsymbol{X})$, then $St(\boldsymbol{X}) \in \boldsymbol{\mathcal{A}}$ trivially follows. Lemma 5.5.20 claims that $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Let $\boldsymbol{Y} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]$, then $\boldsymbol{X} \leq_p \boldsymbol{Y}$ follows from the definition of a composition poset.

First we will show that $\boldsymbol{Y}$ is A-reliable. By monotonicity of A and since $\boldsymbol{X}$ is A-reliable, we have $\boldsymbol{X} \leq_p A(\boldsymbol{X}) \leq_p A(\boldsymbol{Y})$. Therefore,

$$\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{Y}).$$

On the other hand, since $St^2(\boldsymbol{X})$ is a fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$:

$$St^2(\boldsymbol{X}) = A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}(St^2(\boldsymbol{X})) = A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]) = A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{Y}).$$

Therefore,

$$\boldsymbol{Y} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!] \leq_p [\![A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{Y}), A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{Y})]\!] = A(\boldsymbol{Y}).$$

Thus $\boldsymbol{Y}$ is A-reliable.

By Lemma 5.5.20 for $\boldsymbol{Y}$, we now know that $St^1(\boldsymbol{Y}) \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$. Since $\boldsymbol{X} \leq_p \boldsymbol{Y}$, this implies

$$St^1(\boldsymbol{Y}) \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}.$$

Therefore, $[\![St^1(\boldsymbol{Y}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]$ is well-defined. Moreover, by monotonicity of A,

$$\begin{aligned} A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(St^1(\boldsymbol{Y})) &= A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{Y}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) \\ &\preceq A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{Y}), \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}]\!]) \\ &= A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}}(St^1(\boldsymbol{Y})) = St^1(\boldsymbol{Y}) \end{aligned}$$

Thus $St^1(\boldsymbol{Y})$ is a pre-fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$. Since $St^1(\boldsymbol{X})$ is the least fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$, this entails $St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y})$. Moreover, by choice of $\boldsymbol{Y}$ and condition 2 in the definition of a composition poset, $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]^{\boldsymbol{\mathcal{U}}} \preceq St^2(\boldsymbol{X})$. Consequently,

$$St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y}) \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq St^2(\boldsymbol{X}),$$

thus $St(\boldsymbol{X}) \in \boldsymbol{\mathcal{A}}$. □

Theorem 5.5.22. *Let $\mathcal{A} = \langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ be an approximation framework. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-reliable, then $St(\boldsymbol{X})$ is well-defined.*

5.5.2 Monotonic Stable Revision Operator

As outlined by Requirement 5, we require the stable revision operator to be a *monotonic* operator on a cpo. However, while A-reliability is sufficient to ensure well-definedness, on its own it does not assure monotonicity of the stable revision operator. Instead, iAFT further restricts the approximants by means of A-prudence. We now proceed to show that this result remains valid for the generalized approximation space. We first rephrase the notion of A-prudence within the generalized context.

Definition 5.5.23 (A-Prudent)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. An approximant $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-prudent if $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^1(\boldsymbol{X})$.*

We break the monotonicity of St down into two components. On the one hand, we establish that St^1 is monotonic when considering A-reliable approximants. On the other hand, we show that St^2 is anti-monotonic for A-reliable and A-prudent approximants. The former follows relatively effortless from the monotonicity of $A^{\boldsymbol{\mathcal{L}}}$.

Proposition 5.5.24. *Let $\boldsymbol{X}, \boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$ be A-reliable. If $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$, then $St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y})$.*

Proof. Analogous to the first part of the proof for Proposition 3.9 by Denecker et al. [36]. By Lemma 5.5.20, we have

$$St^1(\boldsymbol{Y}) \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}.$$

Therefore $[\![St^1(\boldsymbol{Y}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]$ is well-defined. Then by monotonicity of $A^{\boldsymbol{\mathcal{L}}}$:

$$\begin{aligned} A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(St^1(\boldsymbol{Y})) &= A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{Y}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) \\ &\preceq A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{Y}), \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}]\!]) \\ &= A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}}(St^1(\boldsymbol{Y})) = St^1(\boldsymbol{Y}) \end{aligned}$$

Therefore $St^1(\boldsymbol{Y})$ is a pre-fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$. Since $St^1(\boldsymbol{X})$ is the least (pre-)fixpoint, it must hold that

$$St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y}).$$

□

Corollary 5.5.25. *Let $\boldsymbol{X}, \boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$ be A-reliable. If $\boldsymbol{X} \leq_p \boldsymbol{Y}$, then $St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y})$.*

We will use an analogous reasoning to prove the anti-monotonicity of St^2 for A-reliable and A-prudent approximants. However, whereas $St^1(\boldsymbol{Y}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$ was a clear consequence of Lemma 5.5.20, showing that $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^2(\boldsymbol{Y})$ requires much more

effort. In particular, we will first show that $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}$ must have a fixpoint $f \in \mathcal{L}$ such that $f \preceq St^2(\boldsymbol{X})$. Then, by definition of $St^1(\boldsymbol{Y})$, $St^1(\boldsymbol{Y}) \preceq f \preceq St^2(\boldsymbol{X})$. Moreover, since we assume $\boldsymbol{Y}$ is A-prudent, $\boldsymbol{Y}^{\mathcal{L}} \preceq St^1(\boldsymbol{Y})$, hence $\boldsymbol{Y}^{\mathcal{L}} \preceq St^2(\boldsymbol{X})$.

To assure the existence of f, we again apply Markowsky's theorem. In particular, we construct a set $\mathcal{C} \subseteq [\bot, St^2(\boldsymbol{X})]_{\mathcal{L}}$ such that $\langle \mathcal{C}, \preceq \rangle$ is a cpo for which $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}$ is a monotonic and internal operator. Specifically, we define

$$\mathcal{C} = \{l \in [\boldsymbol{X}^{\mathcal{L}}, St^2(\boldsymbol{X})]_{\mathcal{L}} \mid l \preceq St^1(\boldsymbol{Y})\}$$

where the bounds $\boldsymbol{X}^{\mathcal{L}}$ and $St^1(\boldsymbol{Y})$ emerge to ensure that $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}$ is indeed internal on $\mathcal{C}$.

Lemma 5.5.26. *Let $\boldsymbol{X} \in \mathcal{A}$ be A-reliable and let $\boldsymbol{Y} \in \mathcal{A}$ be A-reliable and A-prudent. If $\boldsymbol{X} \leq_p \boldsymbol{Y}$, then $\langle \mathcal{C}, \preceq \rangle$ is a cpo.*

Proof. We have the following inequalities:

$$\begin{aligned} \boldsymbol{X}^{\mathcal{L}} &\preceq St^2(\boldsymbol{X}) && \text{(Lemma 5.5.20)} \\ \boldsymbol{X}^{\mathcal{L}} &\preceq \boldsymbol{Y}^{\mathcal{L}} && (\boldsymbol{X} \leq_p \boldsymbol{Y}) \\ &\preceq St^1(\boldsymbol{Y}) && (\boldsymbol{Y} \text{ is } A\text{-prudent}) \end{aligned}$$

Clearly, by the inequalities above, $\boldsymbol{X}^{\mathcal{L}} \in \mathcal{C}$. Consequently, $\mathcal{C}$ has a least element $\boldsymbol{X}^{\mathcal{L}}$.

Now, consider a non-empty chain $S \subseteq \mathcal{C}$. By Proposition 5.5.6, we know that $\langle [\boldsymbol{X}^{\mathcal{L}}, St^2(\boldsymbol{X})]_{\mathcal{L}}, \preceq \rangle$ is a cpo and

$$\mathrm{lub}_{\mathcal{L}}(S) \in [\boldsymbol{X}^{\mathcal{L}}, St^2(\boldsymbol{X})]_{\mathcal{L}}.$$

Moreover, since for every $s \in S$, $s \preceq St^1(\boldsymbol{Y})$, $St^1(\boldsymbol{Y})$ must be an upper bound of S. Therefore,

$$\mathrm{lub}_{\mathcal{L}}(S) \preceq St^1(\boldsymbol{Y}).$$

Consequently, $\mathrm{lub}_{\mathcal{L}}(S) \in \mathcal{C}$. Thus $\mathcal{C}$ is a cpo. □

Lemma 5.5.27. *Let $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{A}$ be A-reliable. If $\boldsymbol{X} \leq_p \boldsymbol{Y}$, then $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}$ is a monotonic operator internal on $\langle \mathcal{C}, \preceq \rangle$.*

Proof. For every $l \in \mathcal{C}$, we have

$$\begin{aligned} \boldsymbol{X}^{\mathcal{L}} &\preceq A^{\mathcal{L}}(\boldsymbol{X}) = A^{\mathcal{L}}([\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!]) && (\boldsymbol{X} \text{ is } A\text{-reliable}) \\ &\preceq A^{\mathcal{L}}([\![l, \boldsymbol{Y}^{\mathcal{U}}]\!]) = A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}(l) && (A \text{ is monotonic, } \boldsymbol{X}^{\mathcal{L}} \preceq l \text{ and } \boldsymbol{Y}^{\mathcal{U}} \preceq \boldsymbol{X}^{\mathcal{U}}) \end{aligned}$$

$$\preceq A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}(St^1(\boldsymbol{Y})) = St^1(\boldsymbol{Y}) \qquad (A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}} \text{ is monotonic and } l \preceq St^1(\boldsymbol{Y}))$$

Additionally, let us define

$$l^* = \mathrm{glb}_{\mathcal{A}}[l, \top]_{\mathcal{U}}$$

By Proposition 5.5.8, l^* is well-defined and $l \preceq l^*$. Moreover, since $l \in \mathcal{C}$, it must be the case that $l \preceq St^2(\boldsymbol{X})$. Therefore $St^2(\boldsymbol{X}) \in [l, \top]_{\mathcal{U}}$ and thus $l^* \preceq St^2(\boldsymbol{X})$. Analogously, since $l \preceq St^1(\boldsymbol{Y}) \preceq \boldsymbol{Y}^{\mathcal{U}}$, we find $l^* \preceq \boldsymbol{Y}^{\mathcal{U}}$. Then

$$\begin{aligned}
A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}(l) &= A^{\mathcal{L}}([\![l, \boldsymbol{Y}^{\mathcal{U}}]\!]) \\
&\preceq A^{\mathcal{L}}([\![l, l^*]\!]) && (A \text{ is monotonic and } l^* \preceq \boldsymbol{Y}^{\mathcal{U}}) \\
&\preceq A^{\mathcal{U}}([\![l, l^*]\!]) && (A([\![l, l^*]\!]) \text{ is consistent}) \\
&\preceq A^{\mathcal{U}}([\![\boldsymbol{X}^{\mathcal{L}}, St^2(\boldsymbol{X})]\!]) && (A \text{ is monotonic, } \boldsymbol{X}^{\mathcal{L}} \preceq l \text{ and } l^* \preceq St^2(\boldsymbol{X})) \\
&= A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}(St^2(\boldsymbol{X})) = St^2(\boldsymbol{X})
\end{aligned}$$

Therefore, $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}(l) \in \mathcal{C}$.

With Proposition 5.5.5, we have that $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}$ is a monotonic operator internal on the cpo $\langle \mathcal{C}, \preceq \rangle$. □

Proposition 5.5.28. *Let $\boldsymbol{X} \in \mathcal{A}$ be A-reliable and let $\boldsymbol{Y} \in \mathcal{A}$ be A-reliable and A-prudent. If $\boldsymbol{X} \leq_p \boldsymbol{Y}$, then $St^2(\boldsymbol{Y}) \preceq St^2(\boldsymbol{X})$.*

Proof. From the previous lemmas we know that $\langle \mathcal{C}, \preceq \rangle$ is a cpo for which $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}$ is internal and monotonic. Theorem 2.2.9 then implies that the restriction of $A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}$ to $\mathcal{C}$ must have a least fixpoint f. Therefore

$$\begin{aligned}
St^1(\boldsymbol{Y}) &= \mathrm{lfp}(A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}) && (\text{By definition of } St^1(\boldsymbol{Y})) \\
&\preceq f && (f \text{ is a fixpoint of } A^{\mathcal{L}}_{\boldsymbol{Y}^{\mathcal{U}}}) \\
&\preceq St^2(\boldsymbol{X}) && (f \in [\boldsymbol{X}^{\mathcal{L}}, St^2(\boldsymbol{X})]_{\mathcal{L}})
\end{aligned}$$

Note that this entails that $f = St^1(\boldsymbol{Y})$.

Consequently, the A-prudence of $\boldsymbol{Y}$ implies $\boldsymbol{Y}^{\mathcal{L}} \preceq St^2(\boldsymbol{X})$. Therefore $[\![\boldsymbol{Y}^{\mathcal{L}}, St^2(\boldsymbol{X})]\!]$ is defined. Moreover, since $\boldsymbol{X} \leq_p \boldsymbol{Y}$, $\boldsymbol{X}^{\mathcal{L}} \preceq \boldsymbol{Y}^{\mathcal{L}}$. Now it is easy to see, by monotonicity of A, that

$$A^{\mathcal{U}}_{\boldsymbol{Y}^{\mathcal{L}}}(St^2(\boldsymbol{X})) = A^{\mathcal{U}}([\![\boldsymbol{Y}^{\mathcal{L}}, St^2(\boldsymbol{X})]\!])$$

$$\preceq A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!])$$

$$= A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}(St^2(\boldsymbol{X})) = St^2(\boldsymbol{X})$$

Therefore $St^2(\boldsymbol{X})$ is a pre-fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}}$. Since $St^2(\boldsymbol{Y})$ is the least (pre-)fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}}$,

$$St^2(\boldsymbol{Y}) \leq St^2(\boldsymbol{X}).$$

□

Theorem 5.5.29. *Let $\mathcal{A} = \langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ be an approximation framework with approximating operator A, let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ be A-reliable and $\boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$ be A-reliable and A-prudent. If $\boldsymbol{X} \leq_p \boldsymbol{Y}$, then $St(\boldsymbol{X}) \leq_p St(\boldsymbol{Y})$.*

Proof. Since $\boldsymbol{X} \leq_p \boldsymbol{Y}$, by Proposition 5.5.24, $St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y})$. Moreover, by Proposition 5.5.28, $St^2(\boldsymbol{Y}) \preceq St^2(\boldsymbol{X})$. Therefore, by the (anti-)monotonicity properties of $[\![.,.]\!]$, $St(\boldsymbol{X}) = [\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!] \leq_p [\![St^1(\boldsymbol{Y}), St^2(\boldsymbol{Y})]\!] = St(\boldsymbol{Y})$. □

5.5.3 Internal Stable Revision Operator on a Cpo

Up until now we have established that the generalized stable revision operator is well-defined and monotonic for A-reliable and A-prudent approximants. In this subsection we prove the final requirement before we can apply Markowsky's theorem. In particular, we show that the stable revision operator is internal on the poset induced by the set of A-reliable and A-prudent approximants. Moreover, we verify that this induced poset is a cpo, as expected. First we prove an interesting property relating approximants in this cpo to their images according to the approximator and the stable revision operator.

Proposition 5.5.30. *If $A \in Appr(\boldsymbol{\mathcal{A}})$ and $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-reliable and A-prudent, then $\boldsymbol{X} \leq_p A(\boldsymbol{X}) \leq_p St(\boldsymbol{X})$.*

Proof. By assumption, $\boldsymbol{X}$ is A-reliable, hence, $\boldsymbol{X} \leq_p A(\boldsymbol{X})$. Moreover, since $\boldsymbol{X}$ is A-prudent, it is easy to see that by monotonicity of A

$$A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}) \preceq A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) = A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(St^1(\boldsymbol{X})) = St^1(\boldsymbol{X}).$$

On the other hand, by Lemma 5.5.20 $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$, which entails $\boldsymbol{X} \leq_p [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]$. Then by monotonicity of A, it is clear that

$$St^2(\boldsymbol{X}) = A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}(St^2(\boldsymbol{X})) = A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]) \preceq A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}).$$

Thus $A(\boldsymbol{X}) \leq_p St(\boldsymbol{X})$. □

To demonstrate that the stable revision operator is internal, we first prove A-reliability of $St(\boldsymbol{X})$ for A-reliable and A-prudent approximants $\boldsymbol{X}$ as a relatively simple consequence of the monotonicity of A. In Proposition 5.5.32, this A-reliability is then cleverly used to verify A-prudence of $St(\boldsymbol{X})$.

Proposition 5.5.31. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-reliable and A-prudent, then $St(\boldsymbol{X})$ is A-reliable.*

Proof. By Lemma 5.5.20, we know $St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. By monotonicity of $A^{\boldsymbol{\mathcal{L}}}$, we then get

$$\begin{aligned} St^1(\boldsymbol{X}) &= A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(St^1(\boldsymbol{X})) \\ &= A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) \\ &\preceq A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!]) \\ &= A^{\boldsymbol{\mathcal{L}}}(St(\boldsymbol{X})) \end{aligned}$$

Additionally, since $\boldsymbol{X}$ is A-prudent, $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^1(\boldsymbol{X})$. By anti-monotonicity of $A^{\boldsymbol{\mathcal{U}}}$, we find

$$\begin{aligned} A^{\boldsymbol{\mathcal{U}}}(St(\boldsymbol{X})) &= A^{\boldsymbol{\mathcal{U}}}([\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!]) \\ &\preceq A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]) \\ &= A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}(St^2(\boldsymbol{X})) = St^2(\boldsymbol{X}) \end{aligned}$$

Therefore, by the (anti-)monotonicity properties of $[\![.,.]\!]$,

$$St(\boldsymbol{X}) = [\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!] \leq_p [\![A^{\boldsymbol{\mathcal{L}}}(St(\boldsymbol{X})), A^{\boldsymbol{\mathcal{U}}}(St(\boldsymbol{X}))]\!] = A(St(\boldsymbol{X})).$$

Thus $St(\boldsymbol{X})$ is A-reliable □

Proposition 5.5.32. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-reliable and A-prudent, then $St(\boldsymbol{X})$ is A-prudent.*

Proof. By Proposition 5.5.30 $\boldsymbol{X} \leq_p St(\boldsymbol{X})$. Moreover, by Proposition 5.5.31, $St(\boldsymbol{X})$ is A-reliable. Therefore by Proposition 5.5.24 $St^1(\boldsymbol{X}) \preceq St^1(St(\boldsymbol{X}))$, thus

$$\begin{aligned} St^1(\boldsymbol{X}) &\preceq St^1(St(\boldsymbol{X})) \\ &= A^{\boldsymbol{\mathcal{L}}}_{St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}}(St^1(St(\boldsymbol{X}))) \\ &= A^{\boldsymbol{\mathcal{L}}}([\![St^1(St(\boldsymbol{X})), St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}]\!]) \end{aligned}$$

By condition 2 in the definition of the composition poset, $St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}} \preceq St^2(\boldsymbol{X})$. Moreover, since $\boldsymbol{X}$ is A-prudent, $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^1(\boldsymbol{X}) \preceq St^1(St(\boldsymbol{X}))$. Therefore, by anti-monotonicity of $A^{\boldsymbol{\mathcal{U}}}$

$$\begin{aligned} A^{\boldsymbol{\mathcal{U}}}([\![St^1(St(\boldsymbol{X})), St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}]\!]) &\preceq A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]) \\ &= A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}(St^2(\boldsymbol{X})) = St^2(\boldsymbol{X}) \end{aligned}$$

Therefore, $St(\boldsymbol{X}) = [\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!] \leq_p A([\![St^1(St(\boldsymbol{X})), St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}]\!])$. Thus,

$$\begin{aligned} St(\boldsymbol{X})^{\boldsymbol{\mathcal{L}}} &\preceq A^{\boldsymbol{\mathcal{L}}}([\![St^1(St(\boldsymbol{X})), St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}]\!]) \\ &= A^{\boldsymbol{\mathcal{L}}}_{St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}}(St^1(St(\boldsymbol{X}))) = St^1(St(\boldsymbol{X})) \end{aligned}$$

Thus $St(\boldsymbol{X})$ is A-prudent. □

We have now successfully established that the stable revision operator is indeed internal on the poset induced by the A-reliable and A-prudent approximants. All that remains to be shown is that this induced poset is in fact a cpo. It is trivial to see that the least precise approximant $[\![\bot, \top]\!]$ is both A-reliable and A-prudent, therefore the induced poset has a least element. Hence, it is sufficient to show that the least upper bound in $\boldsymbol{\mathcal{A}}$ for every *non-empty* chain $C \subseteq \boldsymbol{\mathcal{A}}$ of A-reliable and A-prudent approximants is in turn A-reliable and A-prudent.

Proposition 5.5.33. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $C \subseteq \boldsymbol{\mathcal{A}}$ is a non-empty chain of A-reliable and A-prudent approximants, then $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C)$ is A-reliable and A-prudent.*

Proof. Define $\boldsymbol{X} = \mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C)$. Take an arbitrary $\boldsymbol{Y} \in C$. Since $\boldsymbol{Y}$ is A-reliable and $\boldsymbol{Y} \leq_p \boldsymbol{X}$, by monotonicity of A we have

$$\boldsymbol{Y} \leq_p A(\boldsymbol{Y}) \leq_p A(\boldsymbol{X}).$$

By the fact that $\boldsymbol{Y}$ was arbitrarily chosen, $A(\boldsymbol{X})$ must be an upper bound of C, and therefore

$$\boldsymbol{X} \leq_p A(\boldsymbol{X}).$$

Thus $\boldsymbol{X}$ is A-reliable.

On the other hand, by Proposition 5.5.24 and A-prudence of $\boldsymbol{Y}$, we derive that $\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq St^1(\boldsymbol{Y}) \preceq St^1(\boldsymbol{X})$. Therefore, in a similar manner we find that

$$\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}) \preceq St^1(\boldsymbol{X}) = A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]).$$

Moreover, by Lemma 5.5.20, $St^1(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Since $\boldsymbol{X}$ is A-reliable, monotonicity of A then entails

$$A^{\boldsymbol{\mathcal{U}}}([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) = A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}.$$

In summary, we get by the (anti-)monotonicity properties of $[\![.,.]\!]$,

$$[\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!] \leq_p A([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]).$$

Finally, from Proposition 5.4.36, we know that $\boldsymbol{X} = \mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})]\!]$. Moreover, by condition 2 in the definition of a composition poset, $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})$, therefore, by the (anti-)monotonicity properties of $[\![.,.]\!]$,

$$\boldsymbol{X} = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})]\!] \leq_p [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!] \leq_p [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!] = \boldsymbol{X}$$

Therefore $\boldsymbol{X} = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]$. Then it is easy to see that $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) = St^1(\boldsymbol{X})$. Thus $\boldsymbol{X}$ is A-prudent. □

In conclusion, the induced sub-poset of A-reliable and A-prudent approximants is a cpo and the stable revision operator is monotonic and internal on this cpo. This yields the generalized main theorem of AFT.

Theorem 5.5.34. *Let $\mathfrak{A} = \langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ be an approximation framework with approximating operator A. The set of A-reliable and A-prudent elements of $\boldsymbol{\mathcal{A}}$ forms a cpo under the precision order $\leq_p$ with least element $[\![\bot, \top]\!]$. The stable revision operator is a well-defined, increasing and monotonic operator in this poset.*

5.6 Generalization of Consistent AFT

With the generalized concepts introduced in the previous sections, we are now able to define the different types of fixpoints analogously to how they were defined in iAFT. The Kripke-Kleene and Well-founded fixpoints are the least fixpoints of the approximating and stable revision operator, respectively. Their existence follows from Markowsky's theorem.

Definition 5.6.1 (Kripke-Kleene fixpoint)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. The* Kripke-Kleene fixpoint *of A is the least precise fixpoint of A, i.e., $KK(A) = \mathrm{lfp}(A)$.*

Definition 5.6.2 (Stable and Well-founded fixpoints)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. An A-reliable and A-prudent $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is a* stable fixpoint *of A if $\boldsymbol{X}$ is a fixpoint of the stable revision operator St, i.e., $St(\boldsymbol{X}) = \boldsymbol{X}$. The* well-founded fixpoint *of A is the least precise stable fixpoint of A.*

Example 5.6.3 (Example 5.1.3, contd.)**.** *Let us look back at the following simple auto-epistemic theory:*

$$T = \left\{ \begin{array}{l} q \Leftrightarrow \neg Kp. \\ r \Leftrightarrow \neg Kq. \end{array} \right\}$$

In terms of notation, we will use $\bot$ to refer to the least belief state, i.e., the set of all interpretations over $\Sigma = \{p, q, r\}$. Moreover, $\top$ corresponds to the greatest belief state, i.e., the inconsistent belief state $\emptyset$. For all other belief states, we will use a term $\{\varphi\}^\Sigma$ to denote the belief state $\{\mathcal{I}$ is a structure over $\Sigma \mid \mathcal{I} \models \varphi\}$. Note that while each term $\{\varphi\}^\Sigma$ identifies a unique belief state, a belief state will have multiple suitable representations according to this choice of notation.

Recall that the operator O corresponding to this theory is defined such that any belief state $\mathcal{X}$ is mapped to the set $\mathcal{Y}$ of all interpretations $\mathcal{I}$ such that

- *q is true in $\mathcal{I}$ iff Kp is false in $\mathcal{X}$, and*
- *r is true in $\mathcal{I}$ iff Kq is false in $\mathcal{X}$.*

Note that Kp is true in a belief state $\mathcal{X}$ if for every $\mathcal{I}$ in $\mathcal{X}$, p is true in $\mathcal{I}$. Therefore, for any possible belief state $\mathcal{X}$, either Kp is false or it is true, hence by the first rule either q is true for every $\mathcal{I} \in O(\mathcal{X})$, or q is false for every $\mathcal{I} \in O(\mathcal{X})$. Similarly, by the second rule, either r is true for every $\mathcal{I} \in O(\mathcal{X})$, or r is false for every $\mathcal{I} \in O(\mathcal{X})$. Moreover, since there is no objective knowledge about p in the theory, it is not hard to see that for any $\mathcal{I} \in O(\mathcal{X})$, we can take the interpretation $\mathcal{I}'$ such that $\mathcal{I}'$ is $\mathcal{I}$ except that the truth value of p in $\mathcal{I}$ is changed from true to false or vice versa, then $\mathcal{I}'$ must be in $O(\mathcal{X})$ as well. As a result, we can conclude that for any belief state $\mathcal{X}$,

$$O(\mathcal{X}) \in \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma, \{\neg q \wedge r\}^\Sigma, \{\neg q \wedge \neg r\}^\Sigma\}.$$

Let U_i be the ultimate approximator of O for iAFT, i.e., for any interval $\boldsymbol{X}$ of belief states, $U_i(\boldsymbol{X})$ maps $\boldsymbol{X}$ to the smallest interval containing $O^*(\boldsymbol{X}) = \{O(\mathcal{X}) \mid \mathcal{X} \in \boldsymbol{X}\}$. It has been shown that $KK(U_i) = WF(U_i) = (\bot, \top)$, as a consequence of the lack of a smaller interval that approximates the four belief states $\{q \wedge r\}^\Sigma$, $\{q \wedge \neg r\}^\Sigma$, $\{\neg q \wedge r\}^\Sigma$ and $\{\neg q \wedge \neg r\}^\Sigma$. Similarly, we can define U_f as the operator mapping every flower $\boldsymbol{X}$ of belief states to the smallest flower *containing $O^*(\boldsymbol{X})$. With $[\![., .]\!]$ defined as in Example 5.4.2, we now have the following (fastest) monotonic induction of U_f*

$$\begin{aligned} U_f([\![\bot, \{\top\}]\!]) &= [\![\bot, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma, \{\neg q \wedge r\}^\Sigma, \{\neg q \wedge \neg r\}^\Sigma\}]\!] = \boldsymbol{X}_1 \\ &\downarrow \\ U_f(\boldsymbol{X}_1) &= [\![\{q\}^\Sigma, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}]\!] = \boldsymbol{X}_2 \\ &\downarrow \\ U_f(\boldsymbol{X}_2) &= [\![\{q \wedge \neg r\}^\Sigma, \{\{q \wedge \neg r\}^\Sigma\}]\!] = \boldsymbol{X}_3 \\ &\downarrow \end{aligned}$$

$$U_f(\boldsymbol{X}_3) = [\![\{q \wedge \neg r\}^\Sigma, \{\{q \wedge \neg r\}^\Sigma\}]\!] = \boldsymbol{X}_3 = \{\{q \wedge \neg r\}^\Sigma\}$$

Interestingly, whereas for iAFT, the well-founded fixpoint was not strong enough to derive the intended fixpoint, for fAFT, the Kripke-Kleene fixpoint already succeeds in finding the correct solution. In particular, $KK(U_f) = \{\{q \wedge \neg r\}^\Sigma\}$, i.e., the singleton containing the belief state where q is known to be true and r is known to be false.

We expect that the well-founded fixpoint $WF(U_f)$ is at least as precise as $KK(U_f)$. Indeed, in Corollary 5.6.32 we show this for arbitrary approximators. Therefore, since $\boldsymbol{X}_3$ is $\leq_p$-maximal, $WF(U_f)$ must be equal to $\boldsymbol{X}_3$ as well, i.e., $WF(U_f) = \{\{q \wedge r\}^\Sigma\} = KK(U_f)$. In particular, the well-founded fixpoint can be derived using the following (fastest) monotonic induction of St_{U_f}

Iteration 1

$$\begin{aligned} U_f^{\boldsymbol{\mathcal{L}}}([\![\bot, \{\top\}]\!]) &= [\![\bot, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma, \{\neg q \wedge r\}^\Sigma, \{\neg q \wedge \neg r\}^\Sigma\}]\!]^{\boldsymbol{\mathcal{L}}} \\ &= \bot \end{aligned}$$

As a result, $St^1_{U_f}([\![\bot, \{\top\}]\!]) = \bot$.

$$\begin{aligned} U_f^{\boldsymbol{\mathcal{U}}}([\![\bot, \{\bot\}]\!]) &= \{\{q \wedge r\}^\Sigma\}^{\boldsymbol{\mathcal{U}}} = \{\{q \wedge r\}^\Sigma\} = u_1 \\ &\downarrow \\ U_f^{\boldsymbol{\mathcal{U}}}([\![\bot, u_1]\!]) &= [\![\{q\}^\Sigma, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}]\!]^{\boldsymbol{\mathcal{U}}} \\ &= \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\} = u_2 \\ &\downarrow \\ U_f^{\boldsymbol{\mathcal{U}}}([\![\bot, u_2]\!]) &= [\![\{q\}^\Sigma, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}]\!]^{\boldsymbol{\mathcal{U}}} = u_2 \end{aligned}$$

Therefore, $St^2_{U_f}([\![\bot, \{\top\}]\!]) = u_2 = \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}$. Thus, $St_{U_f} = [\![\bot, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}]\!]$.

Iteration 2

$$\begin{aligned} U_f^{\boldsymbol{\mathcal{L}}}([\![\bot, u_2]\!]) &= [\![\{q\}^\Sigma, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}]\!]^{\boldsymbol{\mathcal{L}}} = \{q\}^\Sigma = l_1 \\ &\downarrow \\ U_f^{\boldsymbol{\mathcal{L}}}([\![l_1, u_2]\!]) &= [\![\{q \wedge \neg r\}^\Sigma, \{\{q \wedge \neg r\}^\Sigma\}]\!]^{\boldsymbol{\mathcal{L}}} = \{q \wedge \neg r\}^\Sigma = l_2 \\ &\downarrow \\ U_f^{\boldsymbol{\mathcal{L}}}([\![l_2, u_2]\!]) &= U_f^{\boldsymbol{\mathcal{L}}}([\![\{q \wedge \neg r\}^\Sigma, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}]\!]) \\ &= U_f^{\boldsymbol{\mathcal{L}}}(\{\{q \wedge \neg r\}^\Sigma\}) \\ &= \{\{q \wedge \neg r\}^\Sigma\}^{\boldsymbol{\mathcal{L}}} = \{q \wedge \neg r\}^\Sigma = l_2 \end{aligned}$$

Consequently, $St^1_{U_f}([\![\bot, u_2]\!]) = l_2 = \{q \wedge \neg r\}^\Sigma$. *Moreover, since* $St^2_{U_f}(\boldsymbol{X})$ *only depends on* $\boldsymbol{X}^{\mathcal{L}}$, *and* $[\![\bot, \{\top\}]\!]^{\mathcal{L}} = \bot = [\![\bot, u_2]\!]^{\mathcal{L}}$, *we find that* $St^2_{U_f}([\![\bot, u_2]\!]) = St^2_{U_f}([\![\bot, \{\top\}]\!]) = u_2$. *Hence* $St_{U_f}([\![\bot, u_2]\!]) = [\![l_2, u_2]\!] = [\![\{q \wedge \neg r\}^\Sigma, \{\{q \wedge r\}^\Sigma, \{q \wedge \neg r\}^\Sigma\}]\!] = \{\{q \wedge \neg r\}^\Sigma\} = KK(U_f)$.

It is not hard to see that a third iteration would determine that $\{\{q \wedge \neg r\}^\Sigma\}$ *is a fixpoint of the stable revision operator, consequently,* $WF(U_f) = \{\{q \wedge \neg r\}^\Sigma\}$. *In summary, the fastest monotonic induction of the stable revision operator already finds the intended fixpoint after two iterations.*

Whereas in the previous example, the KK-fixpoint and the well-founded fixpoint coincide, this does not hold in general as demonstrated by the following simple logic program.

Example 5.6.4. *Consider the following simple monotonic logic program which states that* p *is true if* p *is true.*

$$p \leftarrow p.$$

This corresponds to the semantic operator O *where*

$$O := \begin{cases} \emptyset \mapsto \emptyset \\ \{p\} \mapsto \{p\} \end{cases}$$

Clearly, the interpretation $\{p\}$ *is a fixpoint of this operator. Therefore, the* KK*-fixpoint of* O *in any approximation space should always approximate the interpretation* $\{p\}$. *We will show this more generally in Proposition 5.6.29.*

However, the truth of p *is self-supported. As a result, we expect that the well-founded fixpoint of* O *does not include the interpretation* $\{p\}$ *as long as the approximation space and approximator are precise enough to exclude it. For example, when using iAFT (and fAFT*[1]*) with the ultimate approximator* U_O *we derive the Herbrand interpretation* $\emptyset$ *as the well-founded model.*

The sets of supported models and stable models are defined in terms of the *exact fixpoints* of the approximator and the stable revision operator, respectively. However, while previously, in iAFT, exactness was a rigidly defined notion, the generalization of this idea is more open-ended. In fact, whereas for iAFT the approximation space was explicitly defined in terms of the exact space, up until this point, we have never even linked the generalized approximation framework or approximation space to an ordered

[1]Note that for this simple program the approximation spaces in iAFT and fAFT coincide. As a result, the same observations hold for fAFT.

exact space $\mathcal{E}$. Likewise, we have never connected the approximator to an 'original' semantic operator on this exact space.

To define a notion of exactness for $\mathcal{A}$ we first need to determine an 'approximates'-relation between approximants and elements in a poset $\mathcal{E} = \langle \mathcal{E}, \leq \rangle$. In the case of iAFT or fAFT, we know that an approximant $\boldsymbol{X} \in \mathcal{A}$ approximates an element $x \in \mathcal{E}$ if $x \in \boldsymbol{X}$. In general, given an approximant $\boldsymbol{X} \in \mathcal{A}$ and an element $x \in \mathcal{E}$, we will denote '$\boldsymbol{X}$ approximates x' by $\boldsymbol{X} \sim x$. Furthermore, we use the notation $\boldsymbol{X}^{\mathcal{E}}$ to denote the set of elements $x \in \mathcal{E}$ approximated by $\boldsymbol{X}$. Analogously, we use the notation $x^{\mathcal{A}}$ to denote the set of approximants $\boldsymbol{X} \in \mathcal{A}$ that approximate x. Note that it is impossible to give a general formal definition for the approximates relation, since this depends on the chosen approximation space. However, we hypothesize that for many natural approximation spaces this relation can be easily derived. Crucially, the $\sim$-relation allows us to formally define exactness. From now on, we assume that the relation $\sim$ is fixed and given.

Definition 5.6.5 (Exact approximant)**.** *An approximant $\boldsymbol{X} \in \mathcal{A}$ is* exact *with respect to an exact space $\mathcal{E}$ if there exists an $x \in \mathcal{E}$ such that*

$$[\![\bot, \boldsymbol{X}^{\mathcal{U}}]\!]^{\mathcal{E}} = x^{\downarrow} \text{ and } [\![\boldsymbol{X}^{\mathcal{L}}, \top]\!]^{\mathcal{E}} = x^{\uparrow}.$$

For every $x \in \mathcal{E}$, we define $\mathcal{E}(x^{\mathcal{A}}) = \{\boldsymbol{X} \in x^{\mathcal{A}} \mid \boldsymbol{X} \text{ is exact}\}$.

For example, in fAFT or iAFT, the exact approximants are the singletons. In contrast, the trivial approximation space has no exact elements except for the special case where $\mathcal{E}$ corresponds to a singleton and is thus isomorphic to the approximation space, then the only element in the approximation space must be exact. Now we are finally ready to formally define the supported and stable models.

Definition 5.6.6 (Supported Models)**.** *Let $A \in Appr(\mathcal{A})$. In the context of an exact space $\mathcal{E}$, an element $x \in \mathcal{E}$ is a* supported model *if there exists an $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$ such that $\boldsymbol{X} = A(\boldsymbol{X})$. As before, we use the notation $SUP(A)$ to denote the set of supported models.*

Definition 5.6.7 (Stable Models)**.** *Let $A \in Appr(\mathcal{A})$. In the context of an exact space $\mathcal{E}$, an element $x \in \mathcal{E}$ is a* stable model *if there exists an $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$ such that $\boldsymbol{X}$ is a stable fixpoint. As before, we use the notation $ST(A)$ to denote the set of stable models.*

Definition 5.6.8 (Kripke-Kleene and Well-founded Model)**.** *Let $A \in Appr(\mathcal{A})$. In the context of an exact space $\mathcal{E}$, an element $x \in \mathcal{E}$ is called the* Kripke-Kleene model *if $KK(A) \in \mathcal{E}(x^{\mathcal{A}})$. Similarly, x is called the* well-founded model *if $WF(A) \in \mathcal{E}(x^{\mathcal{A}})$.*

Example 5.6.9 (Example 5.1.3, contd.)**.** *For the simple auto-epistemic theory of this example, the belief state $\{q \wedge \neg r\}^{\Sigma}$ is the Kripke-Kleene and the well-founded model in fAFT with the ultimate operator. Moreover, it is also the only stable and the only supported model. Hence, not only is fAFT expressive enough to solve this example, it is also sufficiently strong such that all four traditional semantics coincide.*

While it is essential that a conservative extension of consistent AFT provide the traditional four semantics associated with the different types of fixpoints, it is arguably even more important that these semantics adhere to the principles and ideas that typically characterize them. For example, we expect that any stable model still points to a minimal fixpoint in the exact space. In this regard, we start by making the link between on the one hand a semantic operator on an exact space, and on the other hand an approximator on an approximation space, more explicit in Section 5.6.1. Afterwards, we take a closer look at some of the interesting properties of the different types of fixpoints that were originally introduced by sAFT and iAFT. In particular, the following sections discuss (1) relations between different types of fixpoints and semantics, e.g., between the Kripke-Kleene and the well-founded fixpoint, and (2) noteworthy properties of stable fixpoints and stable models.

5.6.1 Approximating an Exact Space

In practice, the goal of this generalized version of AFT is still to use an approximator A to approximate a possibly non-monotonic operator O on an exact space $\boldsymbol{\mathcal{E}} = \langle \mathcal{E}, \leq \rangle$. Intuitively, it is clear that we cannot just take any approximation space and link it to the given exact space. Instead we expect the approximation space to be strongly related to the exact space. Therefore, although we believe it is impossible to give a concrete generalized definition of $\sim$, it is sensible to impose constraints on this relation to reinforce the link between the spaces. In particular, we have clear intuitions about how approximants, *ALBs* and *AUBs* should relate to the exact space, which should be reflected in $\sim$. For instance, we expect that if an approximant $\boldsymbol{X}$ is less precise than $\boldsymbol{Y}$, it should at least approximate everything approximated by $\boldsymbol{Y}$. Accordingly, we identify a class of approximation spaces for which $\sim$ captures these intuitions. In the remainder of this chapter $\boldsymbol{\mathcal{E}}$ always refers to a cpo which serves as an exact space.

Definition 5.6.10. *Let $\preceq_{\boldsymbol{\mathcal{E}}}$ be a partial order on $\boldsymbol{\mathcal{L}} \cup \boldsymbol{\mathcal{U}} \cup \boldsymbol{\mathcal{E}}$. We say $\preceq_{\boldsymbol{\mathcal{E}}}$ is a* global approximant order *or, in short, an* approximant order*, if*

- *it extends $\preceq$; for every $b_1, b_2 \in \boldsymbol{\mathcal{L}} \cup \boldsymbol{\mathcal{U}}$: $b_1 \preceq_{\boldsymbol{\mathcal{E}}} b_2$ iff $b_1 \preceq b_2$,*
- *it extends $\leq$; for every $x, y \in \boldsymbol{\mathcal{E}}$: $x \preceq_{\boldsymbol{\mathcal{E}}} y$ iff $x \leq y$, and*

- $\bot_{\mathcal{E}} = \bot_{\mathcal{L}}$.

Strictly speaking, it is not necessary for $\preceq_{\mathcal{E}}$ to form a partial order directly on $\mathcal{L} \cup \mathcal{U} \cup \mathcal{E}$. Any poset $\mathcal{E}'$ that is isomorphic to $\mathcal{E}$ would serve just as well, meaning that it suffices if $\preceq_{\mathcal{E}}$ is a partial order on $\mathcal{L} \cup \mathcal{U} \cup \mathcal{E}'$. For the sake of readability, however, we will not emphasize these technical distinctions here and will only return to them when they become relevant.

Definition 5.6.11. *Let $\sim \subseteq \mathcal{A} \times \mathcal{E}$ be a relation. We say the approximation space $\mathcal{A}$* approximates *$\mathcal{E}$ (through $\sim$) if there exists a global approximant order $\preceq_{\mathcal{E}}$ on $\mathcal{L} \cup \mathcal{U} \cup \mathcal{E}$ such that*

1. *for every $l \in \mathcal{L}$, $u \in \mathcal{U}$ and $x \in \mathcal{E}$: $[\![l, u]\!] \sim x$ iff $l \preceq_{\mathcal{E}} x \preceq_{\mathcal{E}} u$.*
2. *for every non-empty chain $C \subseteq \mathcal{E}$ and $\boldsymbol{X} \in \mathcal{A}$: if for every $c \in C$, $\boldsymbol{X} \sim x$, then $\boldsymbol{X} \sim \mathrm{lub}_{\mathcal{E}}(C)$.*
3. *for every chain $C \subseteq \mathcal{A}$ and $x \in \mathcal{E}$, if for every $\boldsymbol{X} \in C$, $\boldsymbol{X} \sim x$, then $\mathrm{lub}_{\mathcal{A}}(C) \sim x$.*

If $\mathcal{A}$ approximates $\mathcal{E}$, we say $\mathcal{A}$ is an approximation space of $\mathcal{E}$, and we denote it by $\mathcal{A} \sim \mathcal{E}$.

For example, it is easily verified that if the exact space $\mathcal{E}$ is a complete lattice, $\mathcal{A}_i(\mathcal{E})$ from Example 5.4.26 is an approximation spaces of $\mathcal{E}$ with $\sim$ the inversion of the membership relation. Similarly, if $\mathcal{E}$ is a bounded-complete cpo, then $\mathcal{A}_f(\mathcal{E})$ from Example 5.4.28 is an approximation space of $\mathcal{E}$ as well. Also the trivial approximation space $\mathcal{A}_{triv}$ is an approximation space of $\mathcal{E}$, in particular, consider $\sim = \mathcal{A}_{triv} \times \mathcal{E}$, i.e., the only element in the approximation space approximates all elements in the exact space.

Proposition 5.6.12. *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be exact spaces such that $\mathcal{E}_2$ is an induced subposet of $\mathcal{E}_1$ that coincides for the least upper bounds of its chains. If $\mathcal{A} \sim \mathcal{E}_1$, then $\mathcal{A} \sim \mathcal{E}_2$.*

Proof. If $\mathcal{A} \sim \mathcal{E}_1$, then there must exist a relation $\sim_1 \subseteq \mathcal{A} \times \mathcal{E}_1$ and a global approximant order $\preceq_{\mathcal{E}_1}$ that satisfy all conditions in Definition 5.6.11. Now take $\sim_2 \subseteq \mathcal{A} \times \mathcal{E}_2$ to be the restriction of $\sim_1$ to this subspace. Similarly, take $\preceq_{\mathcal{E}_2}$ the restriction of $\preceq_{\mathcal{E}_1}$ on $\mathcal{L} \cup \mathcal{U} \cup \mathcal{E}_2$. Then conditions 1 and 3 follow trivially. Moreover, as $\mathcal{E}_1$ and $\mathcal{E}_2$ coincide for least upper bounds of chains in $\mathcal{E}_2$, also condition 2 holds. □

The constraints in Definition 5.6.11 suffice to capture the desired properties of approximants. However, the increased flexibility with respect to the suitable approximation spaces has diminished the strong link between the exact space and

the induced subposets of exact and $\leq_p$-maximal approximants. In earlier versions of AFT, these three posets were isomorphic, but this assumption no longer applies. Notably, the loss of this isomorphic nature breaks some of the properties and relations formulated for sAFT or iAFT. However, under sensible conditions, the link between those spaces is sufficiently restored to re-establish them.

In particular, we notice that in most natural approximation spaces, there is a strong link between the lower decomposition space and the exact space. Specifically, usually, the lower decomposition space $\boldsymbol{\mathcal{L}}$ is at least as refined as the exact space $\boldsymbol{\mathcal{E}}$, i.e., $\boldsymbol{\mathcal{E}}$ is a subposet of $\boldsymbol{\mathcal{L}}$ (or of a poset that is isomorphic to $\boldsymbol{\mathcal{L}}$). Such spaces have two important benefits. On the one hand, they ensure that the knowledge encapsulated by the set of approximants that approximate x is compatible, i.e., every set of approximants approximating x has an upper bound combining their knowledge that still approximates x as there is a most precise approximant that approximates x. Furthermore, it ensures that every element $x \in \boldsymbol{\mathcal{E}}$ can be represented by at least one exact approximant. Note that whenever an element in the exact space cannot be precisely represented by an exact approximant, it is impossible to identify it as a supported or stable model, nor can it correspond to a well-founded or Kripke-Kleene model. This suggests a lack of refinedness in the approximation space.

Definition 5.6.13. *We say the approximation space $\boldsymbol{\mathcal{A}}$* extends *$\boldsymbol{\mathcal{E}}$ if $\boldsymbol{\mathcal{E}}$ is an induced subposet of $\boldsymbol{\mathcal{L}}$ (up to isomorphism). If $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}}$, we say $\boldsymbol{\mathcal{A}}$ is a extension of $\boldsymbol{\mathcal{E}}$.*

Proposition 5.6.14. *If $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}}$, then $\boldsymbol{\mathcal{A}}$ approximates $\boldsymbol{\mathcal{E}}$.*

Proof. As $\boldsymbol{\mathcal{E}}$ is an induced subposet of $\boldsymbol{\mathcal{L}}$, we can just take $\preceq$ as the global approximant order. Now we take the relation $\sim$ such that for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ and $x \in \boldsymbol{\mathcal{E}}$, $\boldsymbol{X} \sim x$ iff $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq x \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Then Item 1 is trivially satisfied. Moreover, Item 2 is a direct consequence of the Chain Interlattice Lub Property, while Item 3 follows from the Interlattice Lb Property. □

Proposition 5.6.15. *Let $\boldsymbol{\mathcal{E}_1}$ and $\boldsymbol{\mathcal{E}_2}$ be exact spaces such that $\boldsymbol{\mathcal{E}_2}$ is an induced subposet of $\boldsymbol{\mathcal{E}_1}$ that coincides with $\boldsymbol{\mathcal{E}_1}$ for the least upper bounds of its chains. Let $\boldsymbol{\mathcal{A}}$ be an approximation space. If $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}_1}$, then $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}_2}$.*

Proof. By definition of an induced subposet, if $\boldsymbol{\mathcal{E}_1}$ is an induced subposet of $\boldsymbol{\mathcal{L}}$, then $\boldsymbol{\mathcal{E}_2}$ is an induced subposet of $\boldsymbol{\mathcal{L}}$ as well. Hence $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}_2}$. □

Interestingly, this categorization of approximation spaces entails an embedding of the exact space into the poset of $\leq_p$-maximal approximants ordered by the truth order.

Proposition 5.6.16. *If $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}}$, then the set $x^{\boldsymbol{\mathcal{A}}}$ of approximants that approximate an element $x \in \boldsymbol{\mathcal{E}}$ has a most precise element*

$$\top_{x^{\boldsymbol{\mathcal{A}}}} = [\![x, \mathrm{glb}_{\boldsymbol{\mathcal{U}}}([x, \top]_{\boldsymbol{\mathcal{U}}})]\!]$$

and $\top_{x^{\boldsymbol{\mathcal{A}}}}^{\boldsymbol{\mathcal{L}}} = x$.

Proof. Since $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}}$, for every $x \in \boldsymbol{\mathcal{E}}$, $x \in \boldsymbol{\mathcal{L}}$, hence $[\![x, \mathrm{glb}_{\boldsymbol{\mathcal{U}}}([x, \top]_{\boldsymbol{\mathcal{U}}})]\!]$ is well-defined. That this is also the most precise element in $x^{\boldsymbol{\mathcal{A}}}$ follows straightforwardly from Definition 5.6.11. Clearly, $\boldsymbol{X} \sim x$ by Item 1. Moreover, by the same condition, for every $\boldsymbol{Y} \in x^{\boldsymbol{\mathcal{A}}}$, $\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}} \preceq x \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Therefore, $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}([x, \top]_{\boldsymbol{\mathcal{U}}}) \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$. Hence $\boldsymbol{Y} \leq_p \boldsymbol{X}$. This holds for every $\boldsymbol{Y} \in x^{\boldsymbol{\mathcal{A}}}$, therefore $\boldsymbol{X}$ is the most precise approximant that approximates x.

Clearly $\top_{x^{\boldsymbol{\mathcal{A}}}}^{\boldsymbol{\mathcal{L}}} \preceq x$ as $\top_{x^{\boldsymbol{\mathcal{A}}}} \sim x$. Furthermore, by Item 2 in Definition 5.4.9, we know that $x \preceq \top_{x^{\boldsymbol{\mathcal{A}}}}^{\boldsymbol{\mathcal{L}}}$. Thus, $\top_{x^{\boldsymbol{\mathcal{A}}}}^{\boldsymbol{\mathcal{L}}} = x$. □

Proposition 5.6.17. *If the approximation space $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}}$, then $\psi : \boldsymbol{\mathcal{E}} \hookrightarrow \boldsymbol{\mathcal{A}} : x \mapsto \top_{x^{\boldsymbol{\mathcal{A}}}}$ forms an embedding of $\boldsymbol{\mathcal{E}}$ in $\langle \mathcal{A}, \leq_t \rangle$.*

Proof. As a direct consequence of Proposition 5.6.16, ψ must be injective. Now we only need to show that ψ is monotonic with respect to the truth order, i.e., if $x \leq y$, then $\psi(x) \leq_t \psi(y)$.

By definition $\psi(x) = [\![x, \mathrm{glb}_{\boldsymbol{\mathcal{U}}}([x, \top]_{\boldsymbol{\mathcal{U}}})]\!]$. As $x \leq y$, $[x, \top]_{\boldsymbol{\mathcal{U}}} \supseteq [y, \top]_{\boldsymbol{\mathcal{U}}}$. Consequently, $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}([x, \top]_{\boldsymbol{\mathcal{U}}}) \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}([y, \top]_{\boldsymbol{\mathcal{U}}})$. In summary, $\psi(x) = [\![x, \mathrm{glb}_{\boldsymbol{\mathcal{U}}}([x, \top]_{\boldsymbol{\mathcal{U}}})]\!] \leq_t [\![y, \mathrm{glb}_{\boldsymbol{\mathcal{U}}}([y, \top]_{\boldsymbol{\mathcal{U}}})]\!] = \psi(y)$.

□

Notably, the fact that the latter category of approximation spaces sustains an embedding of the exact space gives rise to a weak transitivity-like property.

Proposition 5.6.18. *Let $\boldsymbol{\mathcal{A}_1}$ and $\boldsymbol{\mathcal{A}_2}$ be approximation spaces such that $\boldsymbol{\mathcal{A}_1}$ extends $\boldsymbol{\mathcal{E}}$. If for every chain $C \subseteq \boldsymbol{\mathcal{E}}$, $\mathrm{lub}_{\langle \mathcal{A}_1, \leq_t \rangle}(\{\top_{c^{\boldsymbol{\mathcal{A}}}} \mid c \in C\}) = \top_{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C)^{\boldsymbol{\mathcal{A}}}}$, then:*

- *If $\boldsymbol{\mathcal{A}_2}$ approximates $\langle \mathcal{A}_1, \leq_t \rangle$, then $\boldsymbol{\mathcal{A}_2}$ approximates $\boldsymbol{\mathcal{E}}$.*
- *If $\boldsymbol{\mathcal{A}_2}$ extends $\langle \mathcal{A}_1, \leq_t \rangle$, then $\boldsymbol{\mathcal{A}_2}$ extends $\boldsymbol{\mathcal{E}}$.*

Proof. This proposition trivially follows from the fact that there is an embedding of $\boldsymbol{\mathcal{E}}$ to $\langle \mathcal{A}_1, \leq_t \rangle$, hence $\boldsymbol{\mathcal{E}}$ is isomorphic to some $\boldsymbol{\mathcal{E}'}$ such that $\boldsymbol{\mathcal{E}'}$ is an induced subposet of $\langle \mathcal{A}_1, \leq_t \rangle$. Moreover, due to the fact that $\top_{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C)^{\boldsymbol{\mathcal{A}}}} = \mathrm{lub}_{\langle \mathcal{A}_1, \leq_t \rangle}(\{\top_{c^{\boldsymbol{\mathcal{A}}}} \mid c \in C\})$, this induced subposet coincides with $\langle \mathcal{A}_1, \leq_t \rangle$ for the least upper bounds of its chains. Hence, this proposition trivially follows from Propositions 5.6.12 and 5.6.15. □

Example 5.6.19. *Let $\boldsymbol{\mathcal{E}}$ be a complete lattice. Clearly, for every $x, y, z \in \boldsymbol{\mathcal{E}}$, if $y \in [x, z]$, then $[x, z] \leq_p [y, y]$ and $[y, y]$ exactly approximates y, i.e., $\top_{y^{\boldsymbol{\mathcal{A}}}} = [y, y]$. Furthermore, for every $x \in \boldsymbol{\mathcal{E}}$, $x \in [\bot, y]$ iff $x \leq y$ and similarly, $x \in [y, \top]$ iff $y \leq x$. In summary, $\boldsymbol{\mathcal{A}_i}(\boldsymbol{\mathcal{E}})$ extends $\boldsymbol{\mathcal{E}}$. Moreover, it is not hard to see by reasoning analogous with that for Lemma 5.4.34, that for any chain of intervals $C \subseteq \langle \mathcal{A}_i(\boldsymbol{\mathcal{E}}), \leq_t \rangle$, $\mathrm{lub}_{\langle \mathcal{A}_i(\boldsymbol{\mathcal{E}}), \leq_t \rangle}(C) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C^{\boldsymbol{\mathcal{L}}}), \mathrm{lub}_{\boldsymbol{\mathcal{U}}}(C^{\boldsymbol{\mathcal{U}}})]\!]$.*

Let $C_{\boldsymbol{\mathcal{E}}} \subseteq \boldsymbol{\mathcal{E}}$ be a chain of exact elements. It then follows that $C_i = \{\top_{x^{\boldsymbol{\mathcal{A}}}} \mid x \in C_{\boldsymbol{\mathcal{E}}}\} = \{[x, x] \mid x \in C_{\boldsymbol{\mathcal{E}}}\}$ must be a chain in $\langle \mathcal{A}_i(\boldsymbol{\mathcal{E}}), \leq_t \rangle$. Hence, $\mathrm{lub}_{\langle \mathcal{A}_i(\boldsymbol{\mathcal{E}}), \leq_t \rangle}(C_i) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C_i^{\boldsymbol{\mathcal{L}}}), \mathrm{lub}_{\boldsymbol{\mathcal{U}}}(C_i^{\boldsymbol{\mathcal{U}}})]\!] = [\![\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}}), \mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})]\!] = [\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}}), \mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})] = \top_{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})^{\boldsymbol{\mathcal{A}}}}$.

In other words, every approximation space (or extension) of the space of intervals ordered by the truth order is in turn an approximation space (or extension, respectively) of the exact space $\boldsymbol{\mathcal{E}}$. Note that $\langle \mathcal{A}_i(\boldsymbol{\mathcal{E}}), \leq_t \rangle$ is also a complete lattice. Hence, this property entails that we can iteratively build more complex approximation spaces using iAFT for the previous approximation space equipped with the truth order.

Example 5.6.20. *Let $\boldsymbol{\mathcal{E}}$ be a bounded-complete cpo. Clearly, $\boldsymbol{\mathcal{A}_f}(\boldsymbol{\mathcal{E}})$ extends $\boldsymbol{\mathcal{E}}$ since for every $x \in \boldsymbol{\mathcal{E}}$ and every flower $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_f}(\boldsymbol{\mathcal{E}})$, if a $x \in \boldsymbol{X}$, then $\boldsymbol{X} \leq_p \{x\}$ and $\{x\}$ exactly approximates x, i.e., $\top_{x^{\boldsymbol{\mathcal{A}}}} = \{x\}$.*

Let $C_{\boldsymbol{\mathcal{E}}} \subseteq \boldsymbol{\mathcal{E}}$ be a chain of exact elements. It then follows that $C_f = \{\top_{x^{\boldsymbol{\mathcal{A}}}} \mid x \in C_{\boldsymbol{\mathcal{E}}}\} = \{\{x\} \mid x \in C_{\boldsymbol{\mathcal{E}}}\}$ must be a chain in $\langle \mathcal{A}_f, \leq_t \rangle$. Hence, $\mathrm{lub}_{\langle \mathcal{A}_f, \leq_t \rangle}(C_f) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{L}_f}}(C_f^{\boldsymbol{\mathcal{L}}}), \mathrm{lub}_{\boldsymbol{\mathcal{U}_f}}(C_f^{\boldsymbol{\mathcal{U}}})]\!]$.

Moreover, $\mathrm{lub}_{\boldsymbol{\mathcal{L}_f}}(C_f^{\boldsymbol{\mathcal{L}}}) = \mathrm{lub}_{\boldsymbol{\mathcal{L}_f}}(\{\mathrm{glb}_{\boldsymbol{\mathcal{E}}}(\{x\}) \mid x \in C_{\boldsymbol{\mathcal{E}}}\}) = \mathrm{lub}_{\boldsymbol{\mathcal{L}_f}}(C) = \mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C)$. Furthermore, $\mathrm{lub}_{\boldsymbol{\mathcal{U}_f}}(C_f^{\boldsymbol{\mathcal{U}}}) = \mathrm{lub}_{\boldsymbol{\mathcal{U}_f}}(\{Max(\,)^{\boldsymbol{\mathcal{E}}}(\{x\}) \mid x \in C_{\boldsymbol{\mathcal{E}}}\}) = \{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C)\}$. Hence $\mathrm{lub}_{\langle \mathcal{A}_f, \leq_t \rangle}(C_f) = [\![\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}}), \{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})\}]\!] = \mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})^{\uparrow} \cap \{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})\}^{\downarrow} = \{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})\} = \top_{\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C_{\boldsymbol{\mathcal{E}}})^{\boldsymbol{\mathcal{A}}}}$.

In other words, similarly to what we saw for intervals, every approximation space (or extension) of the space of flowers ordered by the truth order is in turn an approximation space (or extension, respectively) of the exact space $\boldsymbol{\mathcal{E}}$. Note that $\langle \mathcal{A}_f(\boldsymbol{\mathcal{E}}), \leq_t \rangle$ is also a bounded-complete cpo. Hence, once again, we can iteratively build more complex approximation spaces using fAFT for the previous approximation space equipped with the truth order.

When working with approximation spaces, not all approximants behave in the way one

might intuitively expect. While most approximants naturally reflect partial information about exact elements and fit neatly into the framework, certain edge cases can arise that deserve closer scrutiny. In particular, we identify two kinds of approximants that may cause undesirable or unintuitive behavior.

The first are empty approximants. These are approximants that, despite being internally consistent (their lower and upper bounds are compatible), do not actually approximate any element of the object space. Their presence seems redundant or unnatural. For this reason, it is often more natural to restrict attention to approximation spaces that do not include them.

The second case involves approximants whose downward closure of exact elements is not closed under limits of chains. Since the constructive processes are inherently step-wise, chains provide a natural way to capture the idea of refining information over successive stages. If the downward closure with respect to the exact elements in an approximant fails to be chain-closed, a construction process may converge to an element that is no longer represented by the approximant, undermining the intended correspondence between approximation and limit.

For these reasons, we will typically focus on non-degenerate extensions of an exact space $\mathcal{E}$, which are free of such degenerate approximants and better capture the constructive intuitions we aim to formalize.

Definition 5.6.21. *Let $\mathcal{A} \sim \mathcal{E}$. We say $\mathcal{A}$ is a* non-degenerate *approximation space of $\mathcal{E}$ if*

- *$\mathcal{L}$ is an induced subposet of $\mathcal{E}$ (up to isomorphism).*
- *For every $X \in \mathcal{A}$, $X^{\mathcal{E}\downarrow}$ is closed.*

Note that in any non-degenerate extension of $\mathcal{E}$, the lower decomposition space must be isomorphic to the exact space. This is not true, however, for just any approximation space of $\mathcal{E}$. For example, the trivial approximation space $\mathcal{A}_{triv}$ is non-degenerate, but it is not an extension and its lower decomposition space is generally not isomorphic to the exact space.

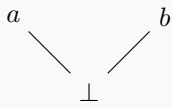
Proposition 5.6.22. *If $\mathcal{A}$ is a non-degenerate approximation space of $\mathcal{E}$, then $\mathcal{A}$ has no empty approximants, i.e., for every $X \in \mathcal{A}$ there exists an $x \in \mathcal{E}$ such that $X \sim x$.*

Proof. Clearly, if $\mathcal{A}$ is a non-degenerate approximation space of $\mathcal{E}$, then for every $X \in \mathcal{A}$, $X^{\mathcal{L}} \in \mathcal{E}$. Furthermore, $X^{\mathcal{L}} \preceq X^{\mathcal{L}} \preceq X^{\mathcal{U}}$. Therefore, $X \sim X^{\mathcal{L}}$. Hence X is not empty. □

All approximation spaces $\mathcal{A}$ we have introduced up until now have been non-degenerate. For example, the approximation spaces in iAFT or fAFT are both non-degenerate.

However, this is not true for every sensible approximation space of $\mathcal{E}$. For instance, to satisfy the conditions posed on an approximation space, sometimes a poset requires the addition of 'virtual elements', i.e., elements without a clear informal meaning with as a sole purpose the satisfaction of those imposed conditions. For example, what if instead of flowers over $\mathcal{E}$, we consider all closed and convex sets over $\mathcal{E}$ to be approximants. In general, the resulting poset does not satisfy all requirements of an approximation space. We can remedy this by adding virtual elements. This is demonstrated in the following example.

Example 5.6.23. *In general, the space of all non-empty closed and convex sets ordered by the superset-order is not an approximation space. Specifically, assuming that it is a composition poset, such a space generally does not satisfy the Interlattice Lb Property. To understand this, let us again consider the following simple exact space:*



Among the closed and convex sets of this space are $\{a, b\}$, $\{a\}$ and $\{b\}$. According to the natural approximates-relation for sets, we have

$$\{a, b\} \sim a \wedge \{a, b\} \sim b$$

$$\{a\} \sim a$$

$$\{b\} \sim b$$

Moreover, we expect $\{a, b\} \leq_p \{a\}$ and $\{a, b\} \leq_p \{b\}$, hence by the conditions of a composition poset

$$\{a, b\}^{\mathcal{L}} \preceq \{a\}^{\mathcal{L}} \preceq \{a\}^{\mathcal{U}} \preceq \{a, b\}^{\mathcal{U}}$$

$$\{a, b\}^{\mathcal{L}} \preceq \{b\}^{\mathcal{L}} \preceq \{b\}^{\mathcal{U}} \preceq \{a, b\}^{\mathcal{U}}$$

Suppose that our space satisfies the Interlattice Lb Property, then there must exist a $u \in \mathcal{U}$ such that $\{a, b\}^{\mathcal{L}} \preceq u \preceq \{a\}^{\mathcal{U}}$ and $u \preceq \{b\}^{\mathcal{U}}$. What would this u represent?

Intuitively we expect that the bound $\{a\}^{\mathcal{U}}$ represents that at most a can be the case, and analogously for b. According to those intuitions, u should identify the

case where only those elements are considered that are below both a and b. On the other hand, we know the ALB $\{a,b\}^{\mathcal{L}}$ should identify all elements that are above either a or b. However, there are no elements in the exact space that satisfy both constraints. Therefore, we get that $[\![\{a,b\}^{\mathcal{L}},u]\!]$ is empty, and thus not a non-empty closed and convex set. Thus, either the Interlattice Lb Property is not satisfied, or $[\![\{a,b\}^{\mathcal{L}},u]\!]$ is undefined, meaning that the space is not a composition poset. Consequently, the described space cannot be an approximation space.

As suggested, we can remedy this by adding virtual elements. Specifically, we add precisely those approximants we are missing in order to satisfy the restrictions of the definition of an approximation space. In this case we need to add u to $\mathcal{U}$ and propagate its effects in $\mathcal{A}$, i.e., by adding the necessary virtual elements for any combination of $l \in \mathcal{L}$ with u such that $l \preceq u$. Specifically, this would entail adding the virtual approximants $[\![\{a,b\}^{\mathcal{L}},u]\!]$ and $[\![\{\bot,a,b\}^{\mathcal{L}},u]\!]$. With these additions, the Interlattice Lb Property is now satisfied and we have obtained an approximation space that includes all non-empty closed convex sets. In summary, if we need the precision provided by convex sets (e.g., when fAFT fails), then we will have to use a non-degenerate approximation space, as these virtual elements are essential inclusions.

Finally, within the context of a semantic operator O on an exact space $\mathcal{E}$, and an approximation space $\mathcal{A}$ of $\mathcal{E}$, not every approximator $A \in Appr(\mathcal{A})$ will match the behavior of O. Instead, just as was done in earlier versions of AFT, we identify a subset of approximators A on $\mathcal{A}$, suitable for O. In this regard, we update the following notions.

Definition 5.6.24. *Let O be an operator on $\mathcal{E}$ and let $\mathcal{A} \sim \mathcal{E}$. We say an approximator $A \in Appr(\mathcal{A})$* approximates *O if for every $\boldsymbol{X} \in \mathcal{A}$ and $x \in \mathcal{E}$, $\boldsymbol{X} \sim x$ implies $A(\boldsymbol{X}) \sim O(x)$.*
We use $Appr_O(\mathcal{A})$ to denote the set of approximators that approximate O.

Definition 5.6.25. *Let O be an operator on $\mathcal{E}$ and let $\mathcal{A} \sim \mathcal{E}$. We say an approximator $A \in Appr_O(\mathcal{A})$* extends *$O$ if for every $x \in \mathcal{E}$, $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$ implies $A(\boldsymbol{X}) \in \mathcal{E}(O(x)^{\mathcal{A}})$.*

Example 5.6.26 (Example 5.1.3, contd.)**.** *Remember that U_f was defined as mapping every flower $\boldsymbol{X}$ of belief states to the smallest flower containing $O^*(\boldsymbol{X}) = \{O(x) \mid x \in \boldsymbol{X}\}$. Clearly, by this definition U_f must approximate O. Moreover, if for some $x \in \mathcal{E}$, $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$, then $\boldsymbol{X} = \{x\}$, thus $O^*(\boldsymbol{X}) = \{O(x)\}$. Clearly, the smallest flower that contains $\{O(x)\}$ is $\{O(x)\}$. Thus, $U_f(\boldsymbol{X}) = \{O(x)\} \in \mathcal{E}(O(x)^{\mathcal{A}})$. In summary, U_f extends O.*

With these connections between approximation spaces and approximators on the one hand, and the exact space and semantic operator on the other hand, we can establish a form of correctness property. Suppose we have two approximators of O defined on two (possibly different) non-degenerate approximation spaces. If one of them yields a model (whether a well-founded or a stable model), then the other will necessarily admit a well-founded fixpoint that approximates that model.

Proposition 5.6.27. *Let O be an operator on $\boldsymbol{\mathcal{E}}$. Let $\boldsymbol{\mathcal{A}_1}$ and $\boldsymbol{\mathcal{A}_2}$ be non-degenerate approximation spaces of $\boldsymbol{\mathcal{E}}$. Let $A \in Appr_O(\boldsymbol{\mathcal{A}_1})$ and let $B \in Appr_O(\boldsymbol{\mathcal{A}_2})$. For every $x \in \boldsymbol{\mathcal{E}}$ such that $x \in ST(A)$, $WF(B) \sim x$.*

Proof. As $x \in ST(A)$, there must exist an exact $\boldsymbol{X}$ such that $\boldsymbol{X} \sim x$ and $\boldsymbol{X} = St_A(\boldsymbol{X})$. Clearly, if $\boldsymbol{X} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$, then $\boldsymbol{X} \sim x$. As A approximates O, $\boldsymbol{X} \leq_p A(\boldsymbol{X}) \sim O(x)$. Since $\boldsymbol{X}$ is exact, this entails $O(x) = x$, i.e., x is a fixpoint of O.

Let $(\boldsymbol{Y}_i)_{i \leq \beta}$ be the fastest monotonic induction of St_B with β an ordinal. We will show by induction that for every $i \leq \beta$, $\boldsymbol{Y}_i \sim x$, then it easily follows that $WF(B) \sim x$ as the well-founded fixpoint is the limit of any terminal monotonic induction. The base case $\boldsymbol{Y}_0$ is trivial since $\boldsymbol{Y}_0$ approximates everything in $\boldsymbol{\mathcal{E}}$. Furthermore, for the case where i is a limit ordinal α, the property easily follows from the induction hypothesis as $\boldsymbol{X}$ must be an upper bound of $\{\boldsymbol{Y}_i \mid i < \alpha\}$. For successor ordinals $i+1$, we look at St_B^1 and St_B^2 separately.

Assume that $(l_j)_{j \leq \gamma}$ is the fastest monotonic induction of $B^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{Y}_i^{\boldsymbol{\mathcal{U}}}}$, with γ an ordinal. Again by induction, we will show for every $j \leq \gamma$ that $l_j \preceq x$. For limit ordinals, this is again a direct consequence of the induction hypothesis. For successor ordinals $j+1$, by the induction hypotheses $\boldsymbol{Y}_i \sim x$ and $l_j \leq x$. Thus $[\![l_j, \boldsymbol{Y}_i^{\boldsymbol{\mathcal{U}}}]\!] \sim x$. Therefore, as B approximates O, $B([\![l_j, \boldsymbol{Y}_i^{\boldsymbol{\mathcal{U}}}]\!]) \sim O(x) = x$. Hence, $B^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{Y}_i^{\boldsymbol{\mathcal{U}}}}(l_j) \leq x$ by condition 1 in Definition 5.6.11.

Let $\boldsymbol{Z} = [\![\boldsymbol{Y}_i^{\boldsymbol{\mathcal{L}}}, St_B^2(\boldsymbol{Y}_i)]\!]$. Let us define $\boldsymbol{\mathcal{P}} = \langle \mathcal{P}, \preceq \rangle$ with $\mathcal{P} = \{y \in \boldsymbol{\mathcal{E}} \mid y \leq x \wedge y \in \boldsymbol{Z}^{\boldsymbol{\mathcal{E}}^{\downarrow}}\}$. $\boldsymbol{\mathcal{P}}$ must be a cpo. Clearly, $\bot \in \mathcal{P}$, hence $\bot$ is the least element of $\boldsymbol{\mathcal{P}}$. Furthermore, as $\boldsymbol{\mathcal{A}_2}$ is non-degenerate, for every non-empty chain $C \subseteq \boldsymbol{\mathcal{P}}$, $\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C) \in \boldsymbol{Z}^{\boldsymbol{\mathcal{E}}^{\downarrow}}$. Furthermore, as x is an upper bound $\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C) \preceq x$.

Furthermore, $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}$ is internal on $\boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{L}_1}$. Let $l \in \boldsymbol{\mathcal{P}}$, then

$$\begin{aligned} A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(l) &= A^{\boldsymbol{\mathcal{L}}}([\![l, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) \\ &\preceq A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}) \\ &= A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}) = \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} = x. \end{aligned}$$

Moreover, we know that $l \in \boldsymbol{Z}^{\boldsymbol{\mathcal{E}}\downarrow}$, thus there must exist a $z \in \boldsymbol{Z}^{\boldsymbol{\mathcal{E}}}$ such that $l \leq z$. By the same reasoning as for Proposition 5.5.21, we know that $\boldsymbol{Z}$ is B-reliable. Therefore, $\boldsymbol{Z} \leq_p B(\boldsymbol{Z}) \sim O(z)$. Then $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X^{\mathcal{U}}}}(l) \preceq_{\boldsymbol{\mathcal{E}_1}} O(z)$. Thus, $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X^{\mathcal{U}}}}(l) \in \boldsymbol{Z}^{\boldsymbol{\mathcal{E}}\downarrow}$. Then, by Markowsky's theorem, $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X^{\mathcal{U}}}}$ must have a fixpoint f in $\boldsymbol{\mathcal{P}}$. As $x = \boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$ is the least fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X^{\mathcal{U}}}}$, $x \leq f$. By definition, also $f \leq x$, thus $f = x$. This entails that $x \in \boldsymbol{\mathcal{P}}$, hence $x \preceq_{\boldsymbol{\mathcal{E}}} \boldsymbol{Z}^{\boldsymbol{\mathcal{U}}} \preceq_{\boldsymbol{\mathcal{E}}} St^2_B(\boldsymbol{Y}_i)$. □

This observation has several important consequences. First, as the approximation space used in iAFT is non-degenerate, we can be confident that any other non-degenerate approximation space will remain consistent with the familiar semantics of iAFT. Second, it guarantees that whenever O is a monotone operator on a cpo, every well-founded fixpoint arising from any operator on a non-degenerate approximation space will approximate the least fixpoint of O.

Corollary 5.6.28. *Let O be a monotonic operator on a cpo $\boldsymbol{\mathcal{E}}$. Let $\boldsymbol{\mathcal{A}} \sim \boldsymbol{\mathcal{E}}$ and let $A \in Appr_O(\boldsymbol{\mathcal{A}})$. If $\boldsymbol{\mathcal{A}}$ is non-degenerate, then $WF(A) \sim \mathrm{lfp}(O)$.*

Proof. Let $x = \mathrm{lfp}(O)$, which exists due to Markowsky's theorem. Consider the tuple $\mathcal{A}_x = \langle \mathcal{L}_x, \mathcal{U}_x, \mathcal{A}_x, \preceq_x, \leq^x_p \rangle$ with

$\boldsymbol{\mathcal{L}_x} = \boldsymbol{\mathcal{E}}$

$\boldsymbol{\mathcal{U}_x} = \{x\} \cup \top$

$\mathcal{A}_x = \{(l, \top) \mid l \in \boldsymbol{\mathcal{E}}\} \cup \{(l, x) \mid l \in \boldsymbol{\mathcal{E}} : l \leq x\}$

$\forall (l_1, u_1), (l_2, u_2) \in \mathcal{A}_x : (l_1, u_1) \leq_p (l_2, u_2)$ iff $l_1 \leq l_2 \wedge (u_1 = \top \vee u_2 = x)$

$\forall (l, u) \in \mathcal{A}_x : [\![l, u]\!] = (l, u)$

$\forall (l, u) \in \mathcal{A}_x : (l, u)^{\boldsymbol{\mathcal{L}}} = (l, u)^1 = l$

$\forall (l, u) \in \mathcal{A}_x : (l, u)^{\boldsymbol{\mathcal{U}}} = (l, u)^2 = u$

It is trivial to see that this is an approximation framework. Furthermore, $\boldsymbol{\mathcal{A}_x}$ is clearly non-degenerate. Take the operator A which maps any approximant $(l, u) \in \boldsymbol{\mathcal{A}_x}$ to

$$\begin{cases} (x, x) & (\text{if } l = x = u) \\ (O(l), \top) & (\text{otherwise}) \end{cases}$$

This operator is monotonic. If $(l_1, u_1) \leq_p (l_2, \top)$, then $l_1 \leq l_2$, therefore as O is monotonic, $O(l_1) \leq O(l_2)$. Furthermore, u_1 must be $\top$ as well. Therefore, $A(l_1, u_1) = (O(l_1), \top) \leq_p (O(l_2), \top) = A(l_2, \top)$. In the other case, if $(l_1, u_1) \leq_p$

(x, x), then $l_1 \leq x$, therefore $O(l_1) \leq O(x) = x$. Hence $A(l_1, u_1) = (O(l_1), \top) \leq_p (x, x) = A(x, x)$. It is also easy to see that $A \sim O$ as O is monotonic. Thus $A \in Appr_O(\boldsymbol{\mathcal{A}})$.

Furthermore, it is not hard to see that any terminal monotonic induction of O can be simulated by a monotonic induction of St. In particular, let $(x_i)_{i\geq 0}$ be a terminal monotonic induction of O. As it is a terminal monotonic induction, it must converge to x. Now, let us take the sequence $((x_i, \top))_{i\geq 0}$. It is trivial to see that $((x_i, \top))_{i\geq 0}$ must be a monotonic induction of A. Moreover, the limit of this sequence must be $(x, \top)$. Furthermore, $St(x, \top) = (x, x)$. With this added step, we have a terminal monotonic induction of A. Hence, x is the well-founded model of A. As, by definition, any well-founded model must also be a stable model, everything now follows from Proposition 5.6.27. □

In contrast to non-degenerate approximation spaces, we are not guaranteed anything about the correctness of approximation spaces with degenerate approximants. To understand this, let us look back at Example 5.6.23. In this example, $[\![\{a, b\}^{\boldsymbol{\mathcal{L}}}, u]\!]$ is $\leq_p$-maximal, but it does not approximate any element of the exact space. Without taking the necessary precautions, the inclusion of such elements might have undesired side-effects. In particular, the connection between the approximator and the semantic operator is formed by relating the behavior of the approximator for an approximant to the behavior of the semantic operator on the elements approximated by said approximant. If there are no such elements, then the behavior of the approximator is completely unregulated.

For example, consider the simple (monotonic) semantic operator on the exact space above where everything is mapped to a. Clearly, the intended fixpoint of this operator is a. However, if we use an approximator that maps the virtual element $[\![\{a, b\}^{\boldsymbol{\mathcal{L}}}, u]\!]$ to itself, and our approximator is precise enough, it will find the well-founded fixpoint $[\![\{a, b\}^{\boldsymbol{\mathcal{L}}}, u]\!]$ which does not approximate our intended fixpoint.

5.6.2 Relating Different Types of Fixpoints

Different types of fixpoints of the approximator correspond to different semantics. Therefore, it is useful to investigate the relations between those different fixpoints, allowing us to make interesting observations about their associated semantics. We show here that the observations made for sAFT and iAFT conservatively generalize to our framework. First, by definition of the Kripke-Kleene and well-founded fixpoints, it is easy to see that they will respectively approximate all supported and stable models.

Proposition 5.6.29. *Let $\boldsymbol{\mathcal{A}} \sim \boldsymbol{\mathcal{E}}$ and let $A \in Appr(\boldsymbol{\mathcal{A}})$.*

- *If $x \in SUP(A)$, then $KK(A) \sim x$.*

- *If* $x \in ST(A)$*, then* $WF(A) \sim x$*.*

Proof. We will only prove the first case, the second is completely analogous. If $x \in SUP(A)$, then by definition of $SUP(A)$ there must exist an exact approximant $\boldsymbol{X} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$ such that $\boldsymbol{X} = A(\boldsymbol{X})$. By definition, $KK(A)$ is the least fixpoint of A, hence $KK(A) \leq_p \boldsymbol{X}$. Finally, by Definition 5.6.11, this entails that $KK(A) \sim x$. □

Consequently, it is equally easy to show that if, respectively, the Kripke-Kleene or well-founded fixpoint is exact, then it identifies a unique supported or stable model.

Proposition 5.6.30. *Let* $A \in Appr(\boldsymbol{\mathcal{A}})$*. For every* $x \in \boldsymbol{\mathcal{E}}$*,*

- *if* $KK(A) \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$*, then* x *is the unique supported model.*
- *if* $WF(A) \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$*, then* x *is the unique stable model.*

Proof. Again, we will only prove the first claim since the second is analogous. Clearly, by definition $KK(A)$ is a fixpoint of A. Furthermore, we assume $KK(A) \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$. Hence, by definition of a supported model, $x \in SUP(A)$. Furthermore, let us assume there exists a $y \in SUP(A)$ different from x, then by Proposition 5.6.29, $KK(A) \sim y$. This contradicts the assumption that $KK(A)$ is exact. Hence x must be the unique supported model. □

Finally, it is also possible to relate the Kripke-Kleene fixpoint to the well-founded fixpoint, and similarly, the set of supported models to the set of stable models. Specifically, it is easy to show that every stable fixpoint is also a fixpoint of A, yielding some interesting results.

Proposition 5.6.31. *Let* $A \in Appr(\boldsymbol{\mathcal{A}})$*. If* $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ *is a fixpoint of* St*, then* $\boldsymbol{X}$ *is a fixpoint of* A*.*

Proof. Since $\boldsymbol{X}$ is a fixpoint of St, $\boldsymbol{X} = St(\boldsymbol{X}) = [\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!]$. Moreover, $St^1(\boldsymbol{X}) = A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!]) = A^{\boldsymbol{\mathcal{L}}}([\![St^1(\boldsymbol{X}), St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}]\!])$. Then, by Proposition 5.4.20, we find $[\![St^1(\boldsymbol{X}), St(\boldsymbol{X})^{\boldsymbol{\mathcal{U}}}]\!] = St(\boldsymbol{X}) = \boldsymbol{X}$. Consequently,

$$St^1(\boldsymbol{X}) = A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}).$$

Following a similar reasoning, we also obtain

$$St^2(\boldsymbol{X}) = A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}).$$

Hence,

$$\boldsymbol{X} = St(\boldsymbol{X}) = [\![St^1(\boldsymbol{X}), St^2(\boldsymbol{X})]\!] = [\![A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}), A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{X})]\!] = A(\boldsymbol{X}).$$

Thus $\boldsymbol{X}$ must be a fixpoint of A. □

Corollary 5.6.32. *Let $A \in Appr(\mathcal{A})$ be an approximator.*

- $KK(A) \leq_p WF(A)$.
- $ST(A) \subseteq SUP(A)$.

5.6.3 Noteworthy Properties of Stable Fixpoints

AFT aims to provide a constructive theory to find the desired $\leq_t$-minimal fixpoint. Crucially, AFT introduced the stable revision operator and the notion of stable fixpoints, and in particular the well-founded fixpoint, to do exactly that. While we have proven the existence of such fixpoint, we have not yet made many promises on its properties. In this section we claim $\leq_t$-minimality of the stable fixpoints, but first we elaborate on the decomposition of a stable fixpoint. Considering the decomposition of a recombined approximant does not always obtain the original *ALB* and *AUB*, it is no longer trivial to see that the lower and upper decomposition of a stable fixpoint are fixpoints of St^1 and St^2 respectively. Nevertheless, it is a useful observation, since it allows us to consider St^1 and St^2 independently for stable fixpoints $\boldsymbol{X}$, i.e., $St^1(\boldsymbol{X})$ holds no additional information about the *AUB* $St^2(\boldsymbol{X})$, or vice versa.

Proposition 5.6.33. *Let $A \in Appr(\mathcal{A})$. If $\boldsymbol{X} \in \mathcal{A}$ is a stable fixpoint of A, then $\boldsymbol{X}^{\mathcal{L}} = St^1(\boldsymbol{X})$ and $\boldsymbol{X}^{\mathcal{U}} = St^2(\boldsymbol{X})$.*

Proof. Since $\boldsymbol{X}$ is A prudent, we have $\boldsymbol{X}^{\mathcal{L}} \preceq St^1(\boldsymbol{X}) \preceq \boldsymbol{X}^{\mathcal{L}}$ where the final inequality follows from condition 2 in the definition of a composition poset. Similarly, by Lemma 5.5.20, we find $\boldsymbol{X}^{\mathcal{U}} \preceq St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\mathcal{U}}$. Hence $\boldsymbol{X}^{\mathcal{L}} = St^1(\boldsymbol{X})$ and $\boldsymbol{X}^{\mathcal{U}} = St^2(\boldsymbol{X})$. □

Now we use the aforementioned property to formally show the desired $\leq_t$-minimality of stable fixpoints. However, note that not every $\leq_t$-minimal fixpoint of A will be a stable fixpoint. This is in line with findings for earlier versions of AFT and with our expectations.

Proposition 5.6.34. *Let $A \in Appr(\mathcal{A})$. If $\boldsymbol{X} \in \mathcal{A}$ is a stable fixpoint, then $\boldsymbol{X}$ is a $\leq_t$-minimal fixpoint of A.*

Proof. Assume there exists a fixpoint $\boldsymbol{Y}$ of A such that $\boldsymbol{Y} \leq_t \boldsymbol{X}$, hence $\boldsymbol{Y}^{\mathcal{L}} \preceq \boldsymbol{X}^{\mathcal{L}}$ and $\boldsymbol{Y}^{\mathcal{U}} \preceq \boldsymbol{X}^{\mathcal{U}}$. Then we have

$$
\begin{aligned}
A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}(\boldsymbol{Y}^{\mathcal{U}}) &= A^{\mathcal{U}}(\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{Y}^{\mathcal{U}}) \\
&\preceq A^{\mathcal{U}}(\boldsymbol{Y}^{\mathcal{L}}, \boldsymbol{Y}^{\mathcal{U}}) \qquad \text{(By anti-monotonicity of } A^{\mathcal{U}} \text{ and since } \boldsymbol{Y}^{\mathcal{L}} \preceq \boldsymbol{X}^{\mathcal{L}})
\end{aligned}
$$

$$= A^{\mathcal{U}}(\boldsymbol{Y}) = \boldsymbol{Y}^{\mathcal{U}} \qquad (\boldsymbol{Y} \text{ is a fixpoint of } A)$$

Therefore, $\boldsymbol{Y}^{\mathcal{U}}$ is a pre-fixpoint of $A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$. Since $\boldsymbol{X}$ is a stable fixpoint, by Proposition 5.6.33, $\boldsymbol{X}^{\mathcal{U}} = St^2(\boldsymbol{X}) = \text{lfp}(A^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}})$. Therefore, $\boldsymbol{X}^{\mathcal{U}} \preceq \boldsymbol{Y}^{\mathcal{U}}$. Consequently, $\boldsymbol{X}^{\mathcal{U}} = \boldsymbol{Y}^{\mathcal{U}}$.

Similarly, we have

$$\begin{aligned} A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}(\boldsymbol{Y}^{\mathcal{L}}) &= A^{\mathcal{L}}(\boldsymbol{Y}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}) \\ &\preceq A^{\mathcal{L}}(\boldsymbol{Y}^{\mathcal{L}}, \boldsymbol{Y}^{\mathcal{U}}) \qquad \text{(By monotonicity of } A^{\mathcal{L}} \text{ and since } \boldsymbol{Y}^{\mathcal{U}} \preceq \boldsymbol{X}^{\mathcal{U}}) \\ &= A^{\mathcal{L}}(\boldsymbol{Y}) = \boldsymbol{Y}^{\mathcal{L}} \qquad (\boldsymbol{Y} \text{ is a fixpoint of } A) \end{aligned}$$

Therefore, $\boldsymbol{Y}^{\mathcal{L}}$ is a pre-fixpoint of $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$. Since $\boldsymbol{X}$ is a stable fixpoint, by Proposition 5.6.33, $\boldsymbol{X}^{\mathcal{L}} = St^1(\boldsymbol{X}) = \text{lfp}(A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}})$. Therefore, $\boldsymbol{X}^{\mathcal{L}} \preceq \boldsymbol{Y}^{\mathcal{L}}$. Consequently, $\boldsymbol{X}^{\mathcal{L}} = \boldsymbol{Y}^{\mathcal{L}}$. In summary, $\boldsymbol{X} = [\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{X}^{\mathcal{U}}]\!] = [\![\boldsymbol{Y}^{\mathcal{L}}, \boldsymbol{Y}^{\mathcal{U}}]\!] = \boldsymbol{Y}$. Therefore, $\boldsymbol{X}$ is $\leq_t$-minimal. □

Proposition 5.6.35. *Let O be an operator on $\boldsymbol{\mathcal{E}}$ and let $\boldsymbol{\mathcal{A}} \sim \boldsymbol{\mathcal{E}}$. Let $A \in Appr_O(\boldsymbol{\mathcal{A}})$. Let $x \in \boldsymbol{\mathcal{E}}$ and let $\boldsymbol{X} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$. If $\boldsymbol{X}$ is a stable fixpoint of A, then x is a $\leq$-minimal fixpoint of O.*

Proof. As $\boldsymbol{X} \sim x$ and $A \in Appr_O(\boldsymbol{\mathcal{A}})$, we know that $A(\boldsymbol{X}) \sim O(x)$. As $\boldsymbol{X}$ is exact, this entails that $x = O(x)$, hence x is a fixpoint of O.

Proof of minimality by contradiction. Let us assume there is a fixpoint $y \in \boldsymbol{\mathcal{E}}$ of O such that $y < x$. Since $\boldsymbol{\mathcal{A}}$ approximates $\boldsymbol{\mathcal{E}}$, this implies $[\![\bot, \boldsymbol{X}^{\mathcal{U}}]\!] \sim y$.

Let us now consider the subposet of $\boldsymbol{\mathcal{L}}$ induced by the set $S = \{l \in \boldsymbol{\mathcal{L}} \mid l \preceq_{\boldsymbol{\mathcal{E}}} y\}$. It is trivial to see that this induced subposet is a cpo.

Moreover, it is not hard to show that $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ is internal on this subposet. In particular, for every $l \in S$, it is clear that $[\![l, \boldsymbol{X}^{\mathcal{U}}]\!] \sim y$. Hence

$$\begin{aligned} & A([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \sim O(y) && (A \text{ approximates } O) \\ \text{implies} \quad & A([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \sim y && (y = O(y)) \\ \text{implies} \quad & A^{\mathcal{L}}([\![l, \boldsymbol{X}^{\mathcal{U}}]\!]) \preceq_{\boldsymbol{\mathcal{E}}} y \\ \text{implies} \quad & A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}(l) \preceq_{\boldsymbol{\mathcal{E}}} y \end{aligned}$$

In summary, S induces a cpo for which $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ is monotonic and internal. Therefore, by Theorem 2.2.9, $A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ must have a fixpoint l^* in S. Since $S \subseteq [\bot, \boldsymbol{X}^{\mathcal{U}}]_{\mathcal{L}}$, by definition of St^1, it then follows that $St^1(\boldsymbol{X}) \preceq l^*$.

As $\boldsymbol{X}$ is a stable fixpoint, by Proposition 5.6.33, $\boldsymbol{X}^{\mathcal{L}} = St^1(\boldsymbol{X}) \preceq l^* \preceq_{\mathcal{E}} y \preceq_{\mathcal{E}} \boldsymbol{X}^{\mathcal{U}}$. Then, since $\mathcal{A}$ approximates $\mathcal{E}$, $\boldsymbol{X} \sim y$. However, by assumption $x \neq y$. This contradicts the assumption that $\boldsymbol{X}$ is exact. □

Finally, the next property claims it is still possible to constructively test whether an element $x \in \mathcal{E}$ is a stable model. Note however, that whereas sAFT or iAFT only have one A-reliable approximant $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$ per $x \in \mathcal{E}$, this is not necessarily true for our generalized framework. Consequently, the constructive test might involve testing multiple approximants.

Proposition 5.6.36. *Let O be an operator on $\mathcal{E}$ and let $\mathcal{A} \sim \mathcal{E}$ be non-degenerate with approximator $A \in Appr_O(\mathcal{A})$. An element $x \in \mathcal{E}$ is a stable model of A iff there exists an $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$ such that $\boldsymbol{X}$ is A-reliable and $\boldsymbol{X}^{\mathcal{L}} = St^1(\boldsymbol{X})$.*

Proof. Clearly, if x is a stable model, then there must exist an approximant $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$ such that $\boldsymbol{X}$ is a stable fixpoint. Hence, by Proposition 5.6.33, $\boldsymbol{X}^{\mathcal{L}} = St^1(\boldsymbol{X})$. Furthermore, by Proposition 5.6.31, $A(\boldsymbol{X}) = \boldsymbol{X}$, therefore $\boldsymbol{X}$ is A-reliable.

Assume $S = \{\boldsymbol{Y} \in \mathcal{A} \mid \boldsymbol{X} \leq_p \boldsymbol{Y}\}$. It is not hard to see that S induces a subposet of $\mathcal{A}$. Moreover, by assumption $\boldsymbol{X}$ is A-reliable, hence A is internal on the induced subposet. Furthermore, by assumption $\boldsymbol{X}^{\mathcal{L}} = St^1(\boldsymbol{X})$. Additionally, $\boldsymbol{X}^{\mathcal{L}} \preceq St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\mathcal{U}}$ by Lemma 5.5.20. As St^1 is monotonic and St^2 is anti-monotonic, St must be internal on said subposet as well. Therefore, St must have a least fixpoint $\boldsymbol{X}'$ in the cpo induced by S. As $\mathcal{A}$ is non-degenerate, we know that $\boldsymbol{X}'$ is not empty, moreover, $\boldsymbol{X} \leq_p \boldsymbol{X}'$ and $\boldsymbol{X} \in \mathcal{E}(x^{\mathcal{A}})$, hence $\boldsymbol{X}' \in \mathcal{E}(x^{\mathcal{A}})$. Therefore $x \in ST(A)$. □

5.7 Well-founded Inductions

To constructively derive the Kripke-Kleene fixpoint of an approximator it suffices to look at monotonic inductions of A. Similarly, for the Well-founded fixpoint, Denecker and Vennekens [30] introduced a similar natural construction process for iAFT, using the notion of *refinements* and *well-founded inductions*. They argue that the latter can be used to give an informal semantics to the semantic operator. Here we show that this work also naturally generalizes to the relaxed definition of approximation spaces. Again, note that our framework is a conservative extension of AFT. As a result, we did not significantly change any of the definitions or properties described here.

Definition 5.7.1 (Refinement)**.** *Let* $A \in Appr(\boldsymbol{\mathcal{A}})$.

- $\boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$ *is an* application refinement *of* $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ *if*

$$\boldsymbol{X} \leq_p \boldsymbol{Y} \leq_p A(\boldsymbol{X}).$$

- $\boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$ *is a* grounding refinement *of* $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ *if there exists an* $u \in \boldsymbol{\mathcal{U}}$ *such that*

$$\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq u \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \text{ and } \boldsymbol{Y} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, u]\!] \text{ and } \boldsymbol{Y} \leq_p A(\boldsymbol{Y}).$$

$\boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$ *is a* refinement *of* $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ *if it is an application refinement of* $\boldsymbol{X}$ *or it is a grounding refinement of* $\boldsymbol{X}$*. For every* $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$*, we use the notations* $R^a_{\boldsymbol{X}}$*,* $R^g_{\boldsymbol{X}}$ *and* $R_{\boldsymbol{X}}$ *to denote the set of its application, grounding and all refinements, respectively.*

Informally, an application refinement corresponds to applying a possibly less precise version of the approximator on our approximant, while a grounding refinement corresponds to deleting an *unfounded set* of elements from our approximant.

Example 5.7.2 (Example 5.1.3, contd.)**.** *Let us take yet another look at the following simple auto-epistemic theory using fAFT:*

$$T = \left\{ \begin{array}{l} q \Leftrightarrow \neg Kp. \\ r \Leftrightarrow \neg Kq. \end{array} \right\}$$

Remember that the operator O *corresponding to this theory is defined such that any belief state* $\mathcal{X}$ *is mapped to the set* $\mathcal{Y}$ *of all interpretations* $\mathcal{I}$ *such that*

- q *is true in* $\mathcal{I}$ *iff* Kp *is false in* $\mathcal{X}$*, and*
- r *is true in* $\mathcal{I}$ *iff* Kq *is false in* $\mathcal{X}$*.*

On the other hand, our approximator U_f *maps every flower* $\boldsymbol{X}$ *of belief states to the smallest* flower *containing* $O^*(\boldsymbol{X}) = \{O(\mathcal{X}) \mid \mathcal{X} \in \boldsymbol{X}\}$.

In the context of AEL, an application refinement deletes belief states from the set which are deemed impossible from every combination of an interpretation and a possible belief state, i.e., there is no possible world in which our agent would have this belief state. On the other hand, a grounding refinement deletes a set of belief states such that no combination of an interpretation and a remaining possible belief states deems them possible.

Specifically for this example this translates as follows. Say we have the set of all belief states, i.e., $\bot_{\mathcal{F}}$. Since nothing is known objectively about p, we see that in any combination of a belief state and an interpretation, the agent will determine that p is not known, i.e., $U_f(\bot_{\mathcal{F}}) = \boldsymbol{X}_{\neg Kp}$ where $\boldsymbol{X}_{\neg Kp}$ is the set of belief states where p is not known. Therefore, all belief states where p is known are always deemed impossible. Hence, as an application refinement of $\bot_{\mathcal{F}}$ we can take any flower that approximates all belief states where p is not known. In such refinement only belief states that are always deemed impossible are deleted.

For example, the flower containing exactly those belief states where p is not known, i.e., $\boldsymbol{X}_{\neg Kp}$, is an application refinement of $\bot_{\mathcal{F}}$, in fact it is the most precise application refinement. Obviously, $\bot_{\mathcal{F}} \supseteq \boldsymbol{X}_{\neg Kp} = U_f(\bot_{\mathcal{F}})$.

As another example, the flower containing exactly those belief states where '$p \wedge q$' is not known denoted by $\boldsymbol{X}_{\neg Kp \wedge q}$, is also an application refinement. The latter flower includes some belief states where p is known on top of those where p is not known, i.e., $\boldsymbol{X}_{\neg Kp \wedge q} \supset \boldsymbol{X}_{\neg Kp}$. Hence, $\bot_{\mathcal{F}} \supseteq \boldsymbol{X}_{\neg Kp \wedge q} \supset \boldsymbol{X}_{\neg Kp} = U_f(\bot_{\mathcal{F}})$.

Note that both these application refinements essentially refine the upper bound, i.e., we consider at most the belief states where p (or for the latter '$p \wedge q$') is not known. Therefore, they are also grounding refinements.

On the other hand, while the set of all belief states where p is not known is the most precise application refinement of $\bot_{\mathcal{F}}$, it is not the most precise grounding refinement. In particular, let us assume we only consider belief states where q is not known to be false, and p is not known. Let us denote this set by $\boldsymbol{X}_{\neg Kq \wedge \neg Kp}$. In this set, since p is never known, q will be known to be true in every possible belief state. Then it is impossible that q is known to be false. Moreover, as argued before, there is no possible world in which the agent would derive that p is known. In other words, once we have restricted ourselves to this set of belief states, it is impossible that we at some point leave the set. Or in symbols, $\boldsymbol{X}_{\neg Kq \wedge \neg Kp} \supseteq A(\boldsymbol{X}_{\neg Kq \wedge \neg Kp})$. Hence, we can safely delete the set of belief states where q is known to be false or p is known. Then, $\boldsymbol{X}_{\neg Kq \wedge \neg Kp}$ is a grounding refinement. Clearly this grounding refinement is more precise than $\boldsymbol{X}_{\neg Kp}$.

Proposition 5.7.3. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is A-reliable, then $\boldsymbol{X} \in R^a_{\boldsymbol{X}} \cap R^g_{\boldsymbol{X}} \subseteq R_{\boldsymbol{X}}$.*

Proof. This is trivial. □

Definition 5.7.4 (Terminal approximant)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}})$ and let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ be A-reliable. $\boldsymbol{X}$ is a* terminal approximant *if $R_{\boldsymbol{X}} = \{\boldsymbol{X}\}$, i.e., if there is no strictly more precise refinement of $\boldsymbol{X}$.*

Analogously, we can conservatively rephrase the notions of a (terminal) well-founded induction as sequences of refinements.

Definition 5.7.5 (Well-founded induction)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. A* well-founded induction *of A is an increasing sequence $(\boldsymbol{X}_i)_{i\leq\beta}$ with $\boldsymbol{X}_i \in \boldsymbol{\mathcal{A}}$, satisfying*

- *$\boldsymbol{X}_{i+1}$ is a refinement of $\boldsymbol{X}_i$, for successor ordinals $i+1 \leq \beta$,*
- *$\boldsymbol{X}_\lambda = \mathrm{lub}_{\boldsymbol{\mathcal{A}}}\{\boldsymbol{X}_i \mid i < \lambda\}$, for limit ordinals $\lambda \leq \beta$.*

Definition 5.7.6 (Terminal well-founded induction)**.** *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. A well-founded induction $(\boldsymbol{X}_i)_{i\leq\beta}$ of A is* terminal *if there does not exist a strictly more precise refinement of its limit, i.e., $\boldsymbol{X}_\beta$ is terminal.*

Example 5.7.7 (Example 5.1.3, contd.)**.** *By definition, a terminal well-founded induction of U_f must start from the least precise approximant, i.e., the set of all possible belief states $\bot_{\mathcal{F}}$:*

$$\boldsymbol{X}_0 = \bot_{\mathcal{F}}.$$

We already discussed that $\boldsymbol{X}_{\neg Kq\wedge\neg Kp}$ is a grounding refinement of $\bot_{\mathcal{F}}$. Thus, let us take

$$\boldsymbol{X}_1 = \boldsymbol{X}_{\neg Kq\wedge\neg Kp}.$$

Finally, we take the application refinement

$$\boldsymbol{X}_2 = U_f(\boldsymbol{X}_1).$$

Note that $\boldsymbol{X}_2$ corresponds to the singleton corresponding to the belief state where q is known to be true and r is known to be false, i.e., the intended fixpoint. Clearly, this is a terminal approximant. Note how this particular terminal well-founded induction mirrors the monotonic induction of the stable revision operator. There are many (slower) alternatives to this specific terminal well-founded induction including the one that parallels the fastest monotonic induction of the approximator U_f. In fact, due to the fact that the Kripke-Kleene fixpoint is already exact, every terminal monotonic induction of the approximator U_f will be a terminal well-founded induction as well. In the upcoming part, we will show that all of these terminal well-founded inductions converge to the same approximant, i.e., $\boldsymbol{X}_2$.

Now all that is left is to show that all terminal well-founded inductions converge to the same limit. In particular, they converge to the well-founded fixpoint. We prove this in three steps. First, we show that every refinement in a well-founded induction is A-reliable and A-prudent. As a result, any well-founded induction corresponds to a chain in the cpo of A-reliable and A-prudent approximants, for which we established that St is monotonic and internal. Second, we demonstrate that any refinement in a

well-founded induction must be less precise than the well-founded fixpoint. Finally, we prove that any terminal approximant must be a fixpoint of the stable revision operator.

5.7.1 A-Reliable and A-Prudent Refinements

Proposition 5.7.8. *Let $A \in Appr(\mathcal{A})$. If $\boldsymbol{X} \in \mathcal{A}$ is A-reliable and $\boldsymbol{Y} \in \mathcal{A}$ is a refinement of $\boldsymbol{X}$, then $\boldsymbol{Y}$ is A-reliable.*

Proof. If $\boldsymbol{Y} \in R^a_{\boldsymbol{X}}$, by definition $\boldsymbol{X} \leq_p \boldsymbol{Y}$, thus by monotonicity of A, $\boldsymbol{Y} \leq_p A(\boldsymbol{X}) \leq_p A(\boldsymbol{Y})$ and $\boldsymbol{Y}$ is A-reliable. For $\boldsymbol{Y} \in R^q_{\boldsymbol{X}}$, by definition $\boldsymbol{Y} \leq_p A(\boldsymbol{Y})$ and $\boldsymbol{Y}$ is again A-reliable. □

Proposition 5.7.9. *Let $A \in Appr(\mathcal{A})$. If $(\boldsymbol{X}_i)_{i \leq \beta}$ is a well-founded induction of A, then for every $i \leq \beta$, $\boldsymbol{X}_i$ is A-reliable.*

Proof. By definition $\boldsymbol{X}_0 = [\![\bot, \top]\!] \in \mathcal{A}$. It is trivial that $[\![\bot, \top]\!] \leq_p A([\![\bot, \top]\!])$, thus $\boldsymbol{X}_0 = [\![\bot, \top]\!]$ is A-reliable.

From Proposition 5.7.8 it follows that if some $\boldsymbol{X}_i$ is A-reliable and $\boldsymbol{X}_{i+1}$ is a refinement of $\boldsymbol{X}_i$, then $\boldsymbol{X}_{i+1}$ is also A-reliable.

Finally, for the limit case $\boldsymbol{X}_\lambda$, we know that by definition $\boldsymbol{X}_i \leq_p \boldsymbol{X}_\lambda$ for every $i < \lambda$, therefore by monotonicity of A, $A(\boldsymbol{X}_i) \leq_p A(\boldsymbol{X}_\lambda)$. Moreover, since $\boldsymbol{X}_i$ is A-reliable by the induction hypothesis, we know that $\boldsymbol{X}_i \leq_p A(\boldsymbol{X}_i)$. Therefore, since $\{\boldsymbol{X}_i \mid i < \lambda\}$ is a chain in $\mathcal{A}$, $\boldsymbol{X}_\lambda = \mathrm{lub}_{\mathcal{A}}(\{(\boldsymbol{X}_i)_{i \leq \lambda}\}) \leq_p (\mathrm{lub}_{\mathcal{A}}(\{A(\boldsymbol{X}_i) \mid i < \lambda\})) \leq_p A(\boldsymbol{X}_\lambda)$. Therefore $\boldsymbol{X}_\lambda$ is A-reliable. □

Proposition 5.7.10. *Let $A \in Appr(\mathcal{A})$. If $\boldsymbol{X} \in \mathcal{A}$ is A-reliable and A-prudent and $\boldsymbol{Y} \in \mathcal{A}$ is a refinement of $\boldsymbol{X}$, then $\boldsymbol{Y}$ is A-prudent.*

Proof. By Proposition 5.7.8 we know that $\boldsymbol{Y}$ is A-reliable. Assume that $\boldsymbol{Y} \in R^a_{\boldsymbol{X}}$. Since $\boldsymbol{X}$ is A-prudent, we have $\boldsymbol{X}^{\mathcal{L}} \preceq St^1(\boldsymbol{X})$. Then, by monotonicity of A, it must hold that $A^{\mathcal{L}}(\boldsymbol{X}) \preceq A^{\mathcal{L}}([\![St^1(\boldsymbol{X}), \boldsymbol{X}^{\mathcal{U}}]\!]) = A^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}(St^1(\boldsymbol{X})) = St^1(\boldsymbol{X})$. Since $\boldsymbol{X} \leq_p \boldsymbol{Y} \leq_p A(\boldsymbol{X})$, $\boldsymbol{Y}^{\mathcal{L}} \preceq A^{\mathcal{L}}(\boldsymbol{X}) \preceq St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y})$ by Proposition 5.5.24. Thus $\boldsymbol{Y}$ is A-prudent.

Now let us assume $\boldsymbol{Y} \in R^q_{\boldsymbol{X}}$. Since by definition $\boldsymbol{X} \leq_p \boldsymbol{Y}$ and $\boldsymbol{X}$ is A-prudent, it trivially follows that $\boldsymbol{X}^{\mathcal{L}} \preceq St^1(\boldsymbol{X}) \preceq St^1(\boldsymbol{Y})$ again by Proposition 5.5.24. Therefore,

$$\begin{aligned} \boldsymbol{Y}^{\mathcal{L}} &\preceq A^{\mathcal{L}}(\boldsymbol{Y}) = A^{\mathcal{L}}([\![\boldsymbol{X}^{\mathcal{L}}, \boldsymbol{Y}^{\mathcal{U}}]\!]) && (\boldsymbol{Y} \text{ is } A\text{-reliable}) \\ &\preceq A^{\mathcal{L}}([\![St^1(\boldsymbol{Y}), \boldsymbol{Y}^{\mathcal{U}}]\!]) = St^1(\boldsymbol{Y}) && (\text{Monotonicity of } A) \end{aligned}$$

Hence, $\boldsymbol{Y}$ is A-prudent. □

Proposition 5.7.11. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $(\boldsymbol{X}_i)_{i \leq \beta}$ is a well-founded induction of A, then for every $i \leq \beta$, $\boldsymbol{X}_i$ is A-prudent.*

Proof. By definition, $\boldsymbol{X}_0 = [\![\bot, \top]\!] \in \boldsymbol{\mathcal{A}}$. It is trivial that $\boldsymbol{X}_0^{\boldsymbol{\mathcal{L}}} = \bot \preceq St^1([\![\bot, \top]\!]) = St^1(\boldsymbol{X}_0)$, thus $\boldsymbol{X}_0 = [\![\bot, \top]\!]$ is A-prudent.

By Proposition 5.7.9, we know that $\boldsymbol{X}_i$ is A-reliable. Now by Proposition 5.7.10 we know that if $\boldsymbol{X}_i$ is A-prudent and $\boldsymbol{X}_{i+1}$ is a refinement of $\boldsymbol{X}_i$, then $\boldsymbol{X}_{i+1}$ is A-prudent.

Notice that in the case of $\boldsymbol{X}_\lambda$, the set $\{\boldsymbol{X}_i \mid i < \lambda\}$ is a chain in $\boldsymbol{\mathcal{A}}$ by definition of a well-founded induction. Moreover, by the induction hypothesis, for every $i < \lambda$, $\boldsymbol{X}_i$ is A-reliable and A-prudent. Then, by Proposition 5.5.33, $\boldsymbol{X}_\lambda = \mathrm{lub}_{\boldsymbol{\mathcal{A}}}\{\boldsymbol{X}_i \mid i < \lambda\}$ is also A-prudent. □

5.7.2 Well-Founded Induction versus Well-Founded Fixpoint

First, let us take a closer look at the most precise refinements, these are closely related to the approximator and the stable revision operator. We will use these to show that any refinement in a well-founded induction approximates the well-founded fixpoint.

Proposition 5.7.12. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is an A-reliable approximant, then $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(R^a_{\boldsymbol{X}}) = A(\boldsymbol{X})$ is the most precise application refinement of $\boldsymbol{X}$.*

Proof. This is trivial. □

Proposition 5.7.13. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is an A-reliable approximant, then $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(R^g_{\boldsymbol{X}}) = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]$ is the most precise grounding refinement of $\boldsymbol{X}$.*

Proof. We will first show that $[\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!] \in R^g_{\boldsymbol{X}}$. From Lemma 5.5.20 we know that $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Since by definition, $St^2(\boldsymbol{X})$ is a fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$, $A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]) = A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}(St^2(\boldsymbol{X})) = St^2(\boldsymbol{X})$. Also from Lemma 5.5.20, it is trivial to see that $\boldsymbol{X} \leq_p [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]$, therefore since $\boldsymbol{X}$ is A-reliable, $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}) \preceq A^{\boldsymbol{\mathcal{L}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!])$. Thus

$$[\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!] \leq_p A([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]).$$

Thus $[\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!] \in R^g_{\boldsymbol{X}}$.

Let $\boldsymbol{Y} \in R^g_{\boldsymbol{X}}$. Now we will show that $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$ is a pre-fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$. Clearly, since $\boldsymbol{Y}$ is a grounding refinement of $\boldsymbol{X}$, $\boldsymbol{X} \leq_p \boldsymbol{Y} \leq_p A(\boldsymbol{Y})$. This entails $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}(\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}) = A^{\boldsymbol{\mathcal{U}}}([\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}]\!]) = A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{Y}) \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$. Thus $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$ is a pre-fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}$.

Hence, $St^2(\boldsymbol{X}) = \mathrm{lfp}(A^{\boldsymbol{\mathcal{U}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}}) \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}$. Consequently, $\boldsymbol{Y} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}]\!] \leq_p [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]$. Since $\boldsymbol{Y}$ was picked arbitrarily from $R^g_{\boldsymbol{X}}$, and $[\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!] \in R^g_{\boldsymbol{X}}$, it must hold that $\mathrm{lub}_{\boldsymbol{\mathcal{A}}}(R^g_{\boldsymbol{X}}) = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]$. □

Proposition 5.7.14. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $(\boldsymbol{X}_i)_{i\leq\beta}$ is a well-founded induction of A and $\boldsymbol{Y} \in \boldsymbol{\mathcal{A}}$ is a stable fixpoint of A, then for every $i \leq \beta$, $\boldsymbol{X}_i \leq_p \boldsymbol{Y}$.*

Proof. Clearly the least precise approximant $\boldsymbol{X}_0 = [\![\bot, \top]\!] \leq_p \boldsymbol{Y}$. It is also trivial to see that if for every $i < \lambda$, $\boldsymbol{X}_i \leq_p \boldsymbol{Y}$, then $\boldsymbol{X}_\lambda = \mathrm{lub}_{\boldsymbol{\mathcal{A}}}\{\boldsymbol{X}_i \mid i < \lambda\} \leq_p \boldsymbol{Y}$. Thus, it suffices to show that if $\boldsymbol{X}_i \leq_p \boldsymbol{Y}$ and $\boldsymbol{X}_{i+1}$ is a refinement of $\boldsymbol{X}_i$, then $\boldsymbol{X}_{i+1} \leq_p \boldsymbol{Y}$.

Let $\boldsymbol{X}_{i+1} \in R^a_{\boldsymbol{X}_i}$. Then by definition $\boldsymbol{X}_{i+1} \leq_p A(\boldsymbol{X}_i)$. From Proposition 5.5.30 it follows that $\boldsymbol{X}_{i+1} \leq_p A(\boldsymbol{X}_i) \leq_p St(\boldsymbol{X}_i) \leq_p St(\boldsymbol{Y}) = \boldsymbol{Y}$, where the last inequality follows from the monotonicity of St.

Let $\boldsymbol{X}_{i+1} \in R^g_{\boldsymbol{X}_i}$. Since $\boldsymbol{X}_i \leq_p \boldsymbol{Y}$, by Proposition 5.5.28, we know that $St^2(\boldsymbol{Y}) \preceq St^2(\boldsymbol{X}_i)$. Therefore, $\boldsymbol{Y}^{\boldsymbol{\mathcal{U}}} \preceq St^2(\boldsymbol{Y}) \preceq St^2(\boldsymbol{X}_i)$. Since $\boldsymbol{X}_i \leq_p \boldsymbol{Y}$, $\boldsymbol{X}_i^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$. This entails that $\boldsymbol{X}_{i+1} \leq_p \mathrm{lub}(R^g_{\boldsymbol{X}_i}) = [\![\boldsymbol{X}_i^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X}_i)]\!] \leq_p [\![\boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}, \boldsymbol{Y}^{\boldsymbol{\mathcal{U}}}]\!] = \boldsymbol{Y}$. □

5.7.3 Stable Fixpoint as a Terminal Approximant

Proposition 5.7.15. *Let $A \in Appr(\boldsymbol{\mathcal{A}})$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ is an A-reliable and A-prudent terminal approximant, then $\boldsymbol{X}$ is a stable fixpoint of A.*

Proof. Since $\boldsymbol{X}$ is A-reliable, $\boldsymbol{X} \leq_p A(\boldsymbol{X})$. Since $\boldsymbol{X}$ is terminal, there is no strictly more precise application-refinement, therefore it is clear that $\boldsymbol{X} = A(\boldsymbol{X})$, i.e., $\boldsymbol{X}$ is a fixpoint of A. Therefore $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} = A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}) = A^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}}(\boldsymbol{X})$, thus $St^1(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$. Since $\boldsymbol{X}$ is also A-prudent, it follows that also $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq St^1(\boldsymbol{X})$, thus $St^1(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$.

Again, since $\boldsymbol{X}$ is terminal, there is also no strictly more precise grounding refinement, hence it follows that $\boldsymbol{X} = [\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!]$, thus $[\![\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}, St^2(\boldsymbol{X})]\!] \leq_p \boldsymbol{X}$ and therefore $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \preceq St^2(X)$. From Lemma 5.5.20, we also know $St^2(\boldsymbol{X}) \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$, thus $St^2(\boldsymbol{X}) = \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Consequently, $St(\boldsymbol{X}) = \boldsymbol{X}$, thus $\boldsymbol{X}$ is a stable fixpoint. □

The aforementioned results and properties are summarized in the following theorem, indicating confluence of all terminal well-founded inductions.

Theorem 5.7.16. *If $(\boldsymbol{X}_i)_{i\leq\beta}$ is a terminal well-founded induction for A, then $(\boldsymbol{X}_i)_{i\leq\beta}$ converges to the well-founded fixpoint, i.e., the well-founded fixpoint of A can be constructed as the limit of any terminal well-founded induction of A.*

Proof. By definition a well-founded induction $(\boldsymbol{X}_i)_{i \leq \beta}$ is terminal if $\boldsymbol{X}_\beta$ is a terminal approximation. By Proposition 5.7.8 and Proposition 5.7.10, we know that $\boldsymbol{X}_\beta \in \boldsymbol{\mathcal{A}}$ is A-reliable and A-prudent. From Proposition 5.7.15 it then follows that $\boldsymbol{X}_\beta$ is a stable fixpoint. On the other hand, from Proposition 5.7.14, it follows that $\boldsymbol{X}_\beta$ is less precise then any arbitrary stable fixpoint of A. Therefore, $\boldsymbol{X}_\beta$ must be the least precise stable fixpoint of A, i.e., the well-founded fixpoint of A. □

This entails that it does not matter in which order we apply the refinements; all terminal well-founded inductions of A will derive the same limit.

5.8 Hierarchy of Approximation Spaces and Approximation Operators

As is the case for iAFT, we will use an approximator A to approximate a (non-monotonic) operator O on the exact space $\boldsymbol{\mathcal{E}} = \langle \mathcal{E}, \leq \rangle$. However, contrary to iAFT, the approximation space is not a fixed entity given an exact space. In fact, there is a whole range of approximation spaces per exact space. This brings up the question of how to select an appropriate approximation space. This section shows that stronger or *more precise* approximation spaces in general lead to semantics that are more precise. Informally, intervals are less precise than flowers, i.e., flowers will be capable of approximating a set in the exact space more accurately than intervals. We have seen that intervals fail to solve Example 5.1.3. On the other hand, we showed that flowers manage to find the intended well-founded model. Thus, fAFT seems to be stronger or more expressive than iAFT.

Naturally, we expect that an increase in expressiveness comes at a cost. In particular, we anticipate that the improved accuracy goes hand in hand with an increase in computational complexity. While a deep analysis of the complexity is beyond the scope of this work, it is clear that in general, larger approximation spaces and larger decomposition spaces may have longer chains and may therefore result in a higher number of steps in the monotonic inductions of the relevant operators. Furthermore, more precise approximation spaces allow a broader spectrum of approximators with the most notable and expressive ones typically exhibiting higher complexity then their counterparts on the simpler approximation space. This in turn increases the complexity of each step in the induction process. Typically, the latter is the biggest contributor to the increase in complexity. For example, note that in iAFT with an exact space of size n, the longest possible chain in the approximation space or decomposition spaces is of length n and occurs when the exact space is a totally ordered set. It is not hard to see that this is also a worst case scenario for fAFT. Moreover, in this worst case scenario, flowers and intervals coincide. Hence, the worst case complexity of fAFT is exactly the same as for

iAFT if the respective approximators are of the same complexity class. Therefore, the complexity results by Pelov [67] straightforwardly generalize to fAFT. As a result, the increased expressiveness of the ultimate semantics of fAFT comes at a negligible cost in terms of complexity compared to the ultimate semantics of iAFT. In contrast, plenty of problems can be solved with significantly less complex (non-ultimate) approximators in iAFT such as Fitting's three-valued immediate consequence operator Φ_P^{SK} associated with Kleene's three-valued truth-assignment $\mathcal{H}^{SK}$. In comparison, in Section 6.1 we will introduce a similar approximator that takes advantage of the added accuracy while being modular and syntactically defined. For general logic programs P, the latter is significantly more complex then Φ_P^{SK}. In other words, when comparing non-ultimate operators the cost of the improved expressiveness may be significant. Hence, a good approximation space and associated approximator balances expressiveness with computational complexity.

Definition 5.8.1. *Let $\mathbb{A}_1 = \langle \mathcal{L}_1, \mathcal{U}_1, \mathcal{A}_1, \preceq_1, \leq_p^1 \rangle$ and $\mathbb{A}_2 = \langle \mathcal{L}_2, \mathcal{U}_2, \mathcal{A}_2, \preceq_2, \leq_p^2 \rangle$ be approximation frameworks approximating an exact space $\boldsymbol{\mathcal{E}}$. We say $\boldsymbol{\mathcal{A}_2}$ is more precise than $\boldsymbol{\mathcal{A}_1}$, denoted by $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$, if there exists an embedding $\psi : \boldsymbol{\mathcal{A}_1} \to \boldsymbol{\mathcal{A}_2}$ and there exist a monotonic function $\iota : \boldsymbol{\mathcal{A}_2} \to \boldsymbol{\mathcal{A}_1}$ such that*

$$\forall \boldsymbol{X}_1 \in \boldsymbol{\mathcal{A}_1}, \boldsymbol{X}_2 \in \boldsymbol{\mathcal{A}_2} : \psi(\boldsymbol{X}_1) \leq_p^2 \boldsymbol{X}_2 \text{ iff } \boldsymbol{X}_1 \leq_p^1 \iota(\boldsymbol{X}_2)$$

and ψ preserves $\sim$, i.e., let $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_1}$ and let $x \in \boldsymbol{\mathcal{E}}$, $\boldsymbol{X} \sim_1 x$ iff $\psi(\boldsymbol{X}) \sim_2 x$.

Intuitively, the monotonic function ι is used to map an approximant $\boldsymbol{X}_2$ in the more precise approximation space to the most precise approximant $\iota(\boldsymbol{X}_2) \in \boldsymbol{\mathcal{A}_1}$ that approximates all exact elements $y \in \boldsymbol{\mathcal{E}}$ approximated by $\boldsymbol{X}_2$. For example, in the case of flowers and intervals over a complete lattice, it would map a flower $\boldsymbol{X}$ to the interval $[\mathrm{glb}(\boldsymbol{X}), \mathrm{lub}(\boldsymbol{X})]$. To simplify the remaining discussions, we ignore the embedding ψ and treat it as the identity function. Notably, for every approximation space $\boldsymbol{\mathcal{A}_1}$ such that $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$ there will be an isomorphic approximation space $\boldsymbol{\mathcal{A}_1}'$ such that $\boldsymbol{\mathcal{A}_1}' \leq_p \boldsymbol{\mathcal{A}_2}$ where the monotonic function ι commutes and ψ corresponds to the identity function. Therefore, the following propositions, theorems and proofs can be generalized straightforwardly to the more general case with an embedding.

Proposition 5.8.2. *Let $\mathbb{A}_1 = \langle \mathcal{L}_1, \mathcal{U}_1, \mathcal{A}_1, \preceq_1, \leq_p^1 \rangle$ and $\mathbb{A}_2 = \langle \mathcal{L}_2, \mathcal{U}_2, \mathcal{A}_2, \preceq_2, \leq_p^2 \rangle$ be approximation frameworks such that $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$ under monotonic function $\iota : \boldsymbol{\mathcal{A}_2} \to \boldsymbol{\mathcal{A}_1}$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_1}$, then $\boldsymbol{X} = \iota(\boldsymbol{X})$.*

Proof. We know by choice of ι that for every $\boldsymbol{X}_1 \in \boldsymbol{\mathcal{A}_1}$ and $\boldsymbol{X}_2 \in \boldsymbol{\mathcal{A}_2}$, $\boldsymbol{X}_1 \leq_p^2 \boldsymbol{X}_2$ iff $\boldsymbol{X}_1 \leq_p^1 \iota(\boldsymbol{X}_2)$. Therefore, if $\boldsymbol{X}_1 = \boldsymbol{X} = \boldsymbol{X}_2$, then $\boldsymbol{X}_1 \leq_p^2 \boldsymbol{X}_2$, hence $\boldsymbol{X} = \boldsymbol{X}_1 \leq_p^1 \iota(\boldsymbol{X}_2) = \iota(\boldsymbol{X})$. Similarly, if $\boldsymbol{X}_1 = \iota(\boldsymbol{X})$ and $\boldsymbol{X}_2 = \boldsymbol{X}$, then $\boldsymbol{X}_1 = \iota(\boldsymbol{X}) = \iota(\boldsymbol{X}_2)$. Consequently, $\boldsymbol{X}_1 \leq_p^1 \iota(\boldsymbol{X}_2)$, hence $\iota(\boldsymbol{X}) = \boldsymbol{X}_1 \leq_p^2 \boldsymbol{X}_2 = \boldsymbol{X}$. Moreover, since ι is monotonic, this implies $\iota(\boldsymbol{X}) \leq_p^1 \boldsymbol{X}$. In summary, $\boldsymbol{X} = \iota(\boldsymbol{X})$. □

Proposition 5.8.3. *Let $\mathfrak{A}_1 = \langle \mathcal{L}_1, \mathcal{U}_1, \mathcal{A}_1, \preceq_1, \leq_p^1 \rangle$ and $\mathfrak{A}_2 = \langle \mathcal{L}_2, \mathcal{U}_2, \mathcal{A}_2, \preceq_2, \leq_p^2 \rangle$ be approximation frameworks such that $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$ under monotonic function $\iota : \boldsymbol{\mathcal{A}_2} \to \boldsymbol{\mathcal{A}_1}$. If $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_2}$, then $\iota(\boldsymbol{X}) \leq_p^2 \boldsymbol{X}$.*

Proof. We know by choice of ι that for every $\boldsymbol{X}_1 \in \boldsymbol{\mathcal{A}_1}$ and $\boldsymbol{X}_2 \in \boldsymbol{\mathcal{A}_2}$, $\boldsymbol{X}_1 \leq_p^2 \boldsymbol{X}_2$ iff $\boldsymbol{X}_1 \leq_p^1 \iota(\boldsymbol{X}_2)$. Take $\boldsymbol{X}_1 = \iota(\boldsymbol{X})$ and $\boldsymbol{X}_2 = \boldsymbol{X}$. We have that $\iota(\boldsymbol{X}) \leq_p^2 \boldsymbol{X}$ iff $\iota(\boldsymbol{X}) \leq_p^1 \iota(\boldsymbol{X})$. Since $\leq_p^1$ is a partial order and thus reflexive, $\iota(\boldsymbol{X}) \leq_p^1 \iota(\boldsymbol{X})$ is true. Therefore $\iota(\boldsymbol{X}) \leq_p^2 \boldsymbol{X}$. □

Naturally, one would assume that more precise approximation spaces provide more accurate semantics. However, this assertion is not entirely straightforward since the accuracy also strongly depends on the chosen approximator. Just as in iAFT, we can have multiple approximators A that approximate an operator O on the exact space ordered by a precision order.

Definition 5.8.4. *Let $A, B \in Appr(\boldsymbol{\mathcal{A}})$. We say B is more precise than A, denoted by $A \leq_p B$, if for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, $A(\boldsymbol{X}) \leq_p B(\boldsymbol{X})$.*

We adopt the term *ultimate approximator* from iAFT to denote the most precise approximator U of O. However, notably, the existence of an ultimate approximator is no longer guaranteed. Specifically, for U to exist, we require the existence of a 'most precise' approximant $\boldsymbol{Y}$ for each approximant $\boldsymbol{X}$ such that $\boldsymbol{Y}$ approximates all images of O for all exact elements approximated by $\boldsymbol{X}$. The ultimate approximator U then maps $\boldsymbol{X}$ to this uniquely defined $\boldsymbol{Y}$. In general, there might exist multiple precision maximal approximants satisfying this condition. Then, instead of having one ultimate approximator, we have multiple *maximal approximators*.

Definition 5.8.5. *Let $\boldsymbol{\mathcal{A}} \sim \boldsymbol{\mathcal{E}}$. Let O be an operator on $\boldsymbol{\mathcal{E}}$. An approximator $A \in Appr_O(\boldsymbol{\mathcal{A}})$ is* maximal *if there does not exist an operator $A' \in Appr_O(\boldsymbol{\mathcal{A}})$ such that $A <_p A'$, i.e., $A \in Max(Appr_O(\boldsymbol{\mathcal{A}}))$. If such a maximal approximator is unique, then it is called the* ultimate approximator*, and it is denoted by U_O (or U if the semantic operator is clear from the context).*

Interestingly, for an important class of approximation spaces, the ultimate approximator is well-defined.

Proposition 5.8.6. *Let $\boldsymbol{\mathcal{A}}$ be a non-degenerate extension of $\boldsymbol{\mathcal{E}}$. If $\boldsymbol{\mathcal{A}}$ is bounded-complete, then any operator O on $\boldsymbol{\mathcal{E}}$ has an ultimate approximator U on $\boldsymbol{\mathcal{A}}$ where U maps any approximant $\boldsymbol{X}$ to*

$$U(\boldsymbol{X}) = \mathrm{glb}_{\boldsymbol{\mathcal{A}}}(\{\top_{y^{\boldsymbol{\mathcal{A}}}} \mid y \in \boldsymbol{\mathcal{E}} : \exists x \in \boldsymbol{\mathcal{E}} : \boldsymbol{X} \sim x \wedge y = O(x)\})$$

Proof. First we will show that U is well-defined, afterwards we will prove that U must be the most precise approximator of O, i.e., if A is an approximator of O, then $A \leq_p U$.

Since $\boldsymbol{\mathcal{A}}$ is bounded-complete, we know that every non-empty set of approximants must have a greatest lower bound. Hence, we only need to show that for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, $\{\top_{y^{\boldsymbol{\mathcal{A}}}} \mid y \in \boldsymbol{\mathcal{E}} : \exists x \in \boldsymbol{\mathcal{E}} : \boldsymbol{X} \sim x \wedge y = O(x)\}$ is not empty. Since $\boldsymbol{\mathcal{A}}$ is non-degenerate, there must exist an $x \in \boldsymbol{\mathcal{E}}$ such that $\boldsymbol{X} \sim x$. We only consider total operators O, hence $y = O(x) \in \boldsymbol{\mathcal{E}}$. Moreover, by assumption, $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}}$, thus $\top_{y^{\boldsymbol{\mathcal{A}}}}$ must exist and must be in the set. As a result, the set is indeed non-empty.

Take an arbitrary $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$ and $x \in \boldsymbol{\mathcal{E}}$. For every A that approximates O we have by definition that $\boldsymbol{X} \sim x$ implies $A(\boldsymbol{X}) \sim O(x) = y$. Moreover, by assumption, $\boldsymbol{\mathcal{A}}$ extends $\boldsymbol{\mathcal{E}}$, therefore $\top_{y^{\boldsymbol{\mathcal{A}}}} = \mathrm{lub}_{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{E}}(y^{\boldsymbol{\mathcal{A}}}))$ exists and since $A(\boldsymbol{X}) \sim x$, we know $A(\boldsymbol{X}) \leq_p \top_{y^{\boldsymbol{\mathcal{A}}}}$. This is true for every $x \in \boldsymbol{\mathcal{E}}$ such that $\boldsymbol{X} \sim x$, hence $A(\boldsymbol{X}) \leq_p \mathrm{glb}_{\boldsymbol{\mathcal{A}}}(\{\top_{y^{\boldsymbol{\mathcal{A}}}} \mid y \in \boldsymbol{\mathcal{E}} : \exists x \in \boldsymbol{\mathcal{E}} : \boldsymbol{X} \sim x \wedge y = O(x)\} = U(\boldsymbol{X})$. Since $\boldsymbol{X}$ was chosen arbitrarily, this holds for every approximant in $\boldsymbol{\mathcal{A}}$, hence $A \leq_p U$. Also A was chosen arbitrarily from all approximators of O, hence U is the most precise approximator of O. □

Proposition 5.8.7. *Let $\boldsymbol{\mathcal{A}}$ be a non-degenerate extension of $\boldsymbol{\mathcal{E}}$. If $\boldsymbol{\mathcal{L}}$ is bounded-complete and $\boldsymbol{\mathcal{U}}$ is a complete lattice, then any operator O on $\boldsymbol{\mathcal{E}}$ has an ultimate approximator U on $\boldsymbol{\mathcal{A}}$ where U maps any approximant $\boldsymbol{X}$ to*

$$U(\boldsymbol{X}) = [\![\mathrm{glb}_{\boldsymbol{\mathcal{L}}}(S^{\boldsymbol{\mathcal{L}}}), \mathrm{lub}_{\boldsymbol{\mathcal{U}}}(S^{\boldsymbol{\mathcal{U}}})]\!]$$

with $S = \{\top_{y^{\boldsymbol{\mathcal{A}}}} \mid y \in \boldsymbol{\mathcal{E}} : \exists x \in \boldsymbol{\mathcal{E}} : \boldsymbol{X} \sim x \wedge y = O(x)\}$).

Proof. We only need to show that if $\boldsymbol{\mathcal{L}}$ is bounded-complete and $\boldsymbol{\mathcal{U}}$ is a complete cpo, then $\boldsymbol{\mathcal{A}}$ must be bounded-complete as well. We have already argued that S must be non-empty, hence, by Lemma 5.4.34, we then obtain the required definition of U.

Let $S' \subseteq \boldsymbol{\mathcal{A}}$ be non-empty. Then $S'^{\boldsymbol{\mathcal{L}}}$ and $S'^{\boldsymbol{\mathcal{U}}}$ are non-empty as well. Since $\boldsymbol{\mathcal{L}}$ is a bounded-complete cpo, this entails that $\mathrm{glb}_{\boldsymbol{\mathcal{L}}}(S'^{\boldsymbol{\mathcal{L}}})$ exists. Similarly, since $\boldsymbol{\mathcal{U}}$ is a complete lattice, $\mathrm{lub}_{\boldsymbol{\mathcal{U}}}(S'^{\boldsymbol{\mathcal{U}}})$ exists as well. Therefore, again by Lemma 5.4.34, $\mathrm{glb}_{\boldsymbol{\mathcal{A}}}(S')$ exists as well. Since S' was chosen arbitrarily, every non-empty subset of $\boldsymbol{\mathcal{A}}$ must have a greatest lower bound, hence $\boldsymbol{\mathcal{A}}$ is indeed bounded-complete. □

Example 5.8.8. *Let $\boldsymbol{\mathcal{E}}$ be a complete lattice. Let us adhere to the definitions in Example 5.4.21. Since $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{U}}$, $\boldsymbol{\mathcal{L}}$ is a bounded-complete cpo, and $\boldsymbol{\mathcal{U}}$ is a complete lattice. We already mentioned that $\boldsymbol{\mathcal{A}}_i(\boldsymbol{\mathcal{E}})$ is conservative, i.e., every interval is non-empty. Then by the proposition above, a semantic operator O on $\boldsymbol{\mathcal{E}}$*

has an ultimate approximator U_i on $\mathcal{A}_i(\mathcal{E})$ such that

$$\begin{aligned}
U_i(\boldsymbol{X}) &= [\![\mathrm{glb}_{\mathcal{L}}(\{\top_{y^{\mathcal{A}}} \mid \exists x, y \in \mathcal{E} : \boldsymbol{X} \sim x \wedge y = O(x)\}^{\mathcal{L}}),\\
&\qquad \mathrm{lub}_{\mathcal{U}}(\{\top_{y^{\mathcal{A}}} \mid \exists x, y \in \mathcal{E} : \boldsymbol{X} \sim x \wedge y = O(x)\}^{\mathcal{U}})]\!]\\
&= [\mathrm{glb}_{\mathcal{E}}(\{[y, y^{\mathcal{L}}] \mid \exists x, y \in \mathcal{E} : x \in \boldsymbol{X} \wedge y = O(x)\}),\\
&\qquad \mathrm{lub}_{\mathcal{E}}(\{[y, y^{\mathcal{U}}] \mid \exists x, y \in \mathcal{E} : x \in \boldsymbol{X} \wedge y = O(x)\})]\\
&= [\mathrm{glb}_{\mathcal{E}}(\{y \mid \exists x, y \in \mathcal{E} : x \in \boldsymbol{X} \wedge y = O(x)\}),\\
&\qquad \mathrm{lub}_{\mathcal{E}}(\{y \mid \exists x, y \in \mathcal{E} : x \in \boldsymbol{X} \wedge y = O(x)\})]\\
&= [\mathrm{glb}_{\mathcal{E}}(O^*(\boldsymbol{X})), \mathrm{lub}_{\mathcal{E}}(O^*(\boldsymbol{X}))]
\end{aligned}$$

This is exactly the ultimate approximator originally introduced for iAFT. Another commonly used approximator in iAFT for logic programming, is Fitting's three-valued immediate consequence operator Φ_P^{SK} and variations thereof, as discussed in Chapter 4. Clearly, $\Phi_P^{SK} \leq_p U_i$.

Example 5.8.9. *Let $\mathcal{E}$ be a bounded-complete cpo. Let us adhere to the definitions in Example 5.4.2. Since $\mathcal{L} = \mathcal{E}$, $\mathcal{L}$ is a bounded-complete cpo. Moreover, we have already shown that $\mathcal{U}$ is a complete lattice. By definition $\mathcal{A}_f(\mathcal{E})$ is conservative, i.e., every flower is non-empty. Then by proposition Proposition 5.8.6, a semantic operator O on $\mathcal{E}$ has an ultimate approximator U_f on $\mathcal{A}_f(\mathcal{E})$ such that*

$$\begin{aligned}
U_f(\boldsymbol{X}) &= \mathrm{glb}_{\mathcal{A}_f(\mathcal{E})}(\{\top_{y^{\mathcal{A}}} \mid \exists x, y \in \mathcal{E} : \boldsymbol{X} \sim x \wedge y = O(x)\})\\
&= \mathrm{glb}_{\mathcal{A}_f(\mathcal{E})}(\{\{y\} \mid \exists x, y \in \mathcal{E} : x \in \boldsymbol{X} \wedge y = O(x)\})\\
&= \mathrm{glb}_{\mathcal{A}_f(\mathcal{E})}(\{\{O(x)\} \mid x \in \boldsymbol{X}\})\\
&= \mathrm{glb}_{\mathcal{A}_f(\mathcal{E})}(O^*(\boldsymbol{X}))
\end{aligned}$$

Hence, $U_f(\boldsymbol{X})$ corresponds to the smallest flower that contains $O^*(\boldsymbol{X})$. This is exactly the operator we have used to solve Example 5.1.3. In Section 6.4, we will give an example of an alternative approximator Γ_P in fAFT for logic programming that parallels the immediate consequence operator Φ_P^{SK}. In particular, just as Φ_P^{SK}, Γ_P is a syntactical and modular operator. Let Φ_P^f be the natural extension

of Φ_P^{SK} on $\mathcal{F}$ as defined in Section 5.8.2, i.e., it maps every flower to the image of Φ_P^{SK} of the smallest interval that contains it. Naturally, $\Phi_P^f \leq_p \Gamma_P \leq_p U_f$.

5.8.1 Hierarchy in Expressiveness of Approximators

The level of precision of the chosen approximator influences the accuracy of the semantics as is formalized in the following theorems which are generalizations of Theorem 5.2 and 5.3 in the work of Denecker [27].

Lemma 5.8.10 (Corresponding with Lemma 5.1 by Denecker [27])**.** *Let $A, B \in Appr(\mathcal{A})$. If $A \leq_p B$ and $\boldsymbol{X} \in \mathcal{A}$ is A-reliable and A-prudent, then*

- *$\boldsymbol{X}$ is B-reliable and B-prudent.*
- *$St_A(\boldsymbol{X}) \leq_p St_B(\boldsymbol{X})$.*

Proof. Clearly $A(\boldsymbol{X}) \leq_p B(\boldsymbol{X})$. Therefore, if $\boldsymbol{X}$ is A-reliable, then $\boldsymbol{X} \leq_p A(\boldsymbol{X}) \leq_p B(\boldsymbol{X})$, hence $\boldsymbol{X}$ is B-reliable. Moreover, in an analogous way to the proof in [27], we can derive that $St_A^1(\boldsymbol{X}) \preceq St_B^1(\boldsymbol{X})$ and $St_B^2(\boldsymbol{X}) \preceq St_A^2(\boldsymbol{X})$. Therefore, by A-prudence of $\boldsymbol{X}$, $\boldsymbol{X}^{\mathcal{L}} \preceq St_A^1(\boldsymbol{X}) \preceq St_B^1(\boldsymbol{X})$, hence $\boldsymbol{X}$ is B-prudent.

Furthermore, by the (anti-)monotonicity properties of $[\![.,.]\!]$,

$$St_A(\boldsymbol{X}) = [\![St_A^1(\boldsymbol{X}), St_A^2(\boldsymbol{X})]\!] \leq_p [\![St_B^1(\boldsymbol{X}), St_B^2(\boldsymbol{X})]\!] = St_B.$$

□

Theorem 5.8.11 (Corresponding with Theorem 5.2 by Denecker [27])**.** *Let $A, B \in Appr(\mathcal{A})$. If $A \leq_p B$, then*

- *$KK(A) \leq_p KK(B)$.*
- *$WF(A) \leq_p WF(B)$.*

Proof. Since $A \leq_p B$, $A(KK(B)) = A(\mathrm{lfp}(B)) \leq_p B(\mathrm{lfp}(B)) = \mathrm{lfp}(B) = KK(B)$. Thus $KK(B)$ is a pre-fixpoint of A. Since by Theorem 2.2.9, the least fixpoint is also the least pre-fixpoint, $KK(A) = \mathrm{lfp}(A) \leq_p KK(B)$.

On the other hand, let $\{\boldsymbol{X}_i\}_{i \leq \beta}$ be the fastest monotonic induction of St_A. Now we will show that for every $i \leq \beta$, $\boldsymbol{X}_i \leq_p WF(B)$. Then $WF(A) = \mathrm{lfp}(St_A) = \boldsymbol{X}_\beta \leq_p WF(B)$. Clearly $\boldsymbol{X}_0 = [\![\bot, \top]\!] \leq_p WF(B)$. Let us assume that for $i \leq \beta$, $\boldsymbol{X}_i \leq_p WF(B)$. By Lemma 5.8.10 we find for the successor ordinal $i+1$, $\boldsymbol{X}_{i+1} = St_A(\boldsymbol{X}_i) \leq_p St_B(\boldsymbol{X}_i)$. Since, again by Lemma 5.8.10, $\boldsymbol{X}_i$ is B-reliable

and B-prudent, by monotonicity of St_B, we then derive $\boldsymbol{X}_{i+1} \leq_p St_B(\boldsymbol{X}_i) \leq_p St_B(WF(B)) = St_B(\text{lfp}(St_B)) = \text{lfp}(St_B) = WF(B)$. The case of the limit ordinal λ is straightforward. □

Theorem 5.8.12 (Corresponding with part 1 of Theorem 5.3 by Denecker [27])**.** *Let $\boldsymbol{\mathcal{A}} \sim \boldsymbol{\mathcal{E}}$. Let $A, B \in Appr(\boldsymbol{\mathcal{A}})$ be approximators of a semantic operator O on $\boldsymbol{\mathcal{E}}$. If $A \leq_p B$, then*

$$SUP(A) \subseteq SUP(B).$$

Proof. Let $x \in SUP(A)$. Then there must exist an $\boldsymbol{X} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$ such that $\boldsymbol{X} = A(\boldsymbol{X})$. Since $A \leq_p B$, we now know $\boldsymbol{X} = A(\boldsymbol{X}) \leq_p B(\boldsymbol{X})$. Now let us take the subset $\boldsymbol{X}^{\uparrow} \cap \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}}) \subseteq \boldsymbol{\mathcal{A}}$ of all approximants that are more precise than $\boldsymbol{X}$. It is trivial to see that, since $\boldsymbol{\mathcal{A}}$ is a cpo, $\boldsymbol{X}^{\uparrow}$ must induce a cpo as well, with least element $\boldsymbol{X}$. Therefore, by the second condition in Definition 5.6.11, $\boldsymbol{X}^{\uparrow} \cap \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$ must, in turn, induce a cpo with least element $\boldsymbol{X}$.

Additionally, for every $\boldsymbol{Y} \in \boldsymbol{X}^{\uparrow}$, $\boldsymbol{X} \leq_p \boldsymbol{Y}$, hence by monotonicity of B, $\boldsymbol{X} \leq_p B(\boldsymbol{X}) \leq_p B(\boldsymbol{Y})$. Furthermore, if $\boldsymbol{Y} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$, then $\boldsymbol{Y} \sim x$, hence $B(\boldsymbol{Y}) \sim Ox = x$. Then it must be the case that $B(\boldsymbol{Y}) \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$. In summary, B is a monotonic, internal operator on the cpo induced by $\boldsymbol{X}^{\uparrow} \cap \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$, hence, by Theorem 2.2.9, B must have a least fixpoint $\boldsymbol{Y}$ in this cpo. Clearly, $\boldsymbol{Y} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$ and $B(\boldsymbol{Y}) = \boldsymbol{Y}$, thus $x \in SUP(B)$. □

Theorem 5.8.13 (Corresponding with part 2 of Theorem 5.3 by Denecker [27])**.** *Let $\boldsymbol{\mathcal{A}} \sim \boldsymbol{\mathcal{E}}$ be non-degenerate and let O be an operator on $\boldsymbol{\mathcal{E}}$. Let $A, B \in Appr_O(\boldsymbol{\mathcal{A}})$. If $A \leq_p B$, then*

$$ST(A) \subseteq ST(B).$$

Proof. Analogous to the proof of Theorem 5.8.12. Let $x \in ST(A)$. Then there must exist an $\boldsymbol{X} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$ such that $\boldsymbol{X}$ is a fixpoint of St_A. Moreover, $\boldsymbol{X} = St_A(\boldsymbol{X}) \leq_p St_B(\boldsymbol{X})$ by Lemma 5.8.10. Now, let us look at the cpo of B-reliable and B-prudent approximants induced by the subset $\boldsymbol{X}^{\uparrow}$. Clearly $\boldsymbol{X}$ is B-reliable and B-prudent. If $\boldsymbol{Y} \in \boldsymbol{X}^{\uparrow}$, then $\boldsymbol{X} \leq_p \boldsymbol{Y}$, hence by Proposition 5.5.30, $St_B(\boldsymbol{X}) \leq_p St_B(\boldsymbol{Y})$. Thus St_B is a monotonic operator on the cpo induced by $\boldsymbol{X}^{\uparrow}$. Therefore, by Theorem 2.2.9, St_B has a least fixpoint $\boldsymbol{Z}$ in said cpo. As $\boldsymbol{X}$ is exact and $\boldsymbol{\mathcal{A}}$ is non-degenerate, this entails that $\boldsymbol{Z} \in \boldsymbol{\mathcal{E}}(x^{\boldsymbol{\mathcal{A}}})$, hence $x \in ST(B)$.

□

5.8.2 Hierarchy in Expressiveness of Approximation Spaces

While we cannot give a general statement about the relative accuracy of approximating spaces for any approximator, it is clear that the sets of approximators are strongly related. Intuitively, if $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$, any approximator on $\boldsymbol{\mathcal{A}_1}$ can be transformed into an approximator on $\boldsymbol{\mathcal{A}_2}$ and vice versa. From Definition 5.8.1, we know there is a monotonic function $\iota : \boldsymbol{\mathcal{A}_2} \to \boldsymbol{\mathcal{A}_1}$ that preserves the order. Therefore, if A_1 is an approximator on $\boldsymbol{\mathcal{A}_1}$, i.e., a $\leq_p$-monotonic operator on $\boldsymbol{\mathcal{A}_1}$, then the composition[2] $(A_1 \circ \iota)$ is a monotonic operator on $\boldsymbol{\mathcal{A}_2}$ and thus an approximator on $\boldsymbol{\mathcal{A}_2}$. Similarly, if $A_2 \in Appr(\boldsymbol{\mathcal{A}_2})$, then $(\iota \circ A_2)$ is an approximator on $\boldsymbol{\mathcal{A}_1}$. In the remaining part of the section we assume that $\mathbb{A}_1 = \langle \mathcal{L}_1, \mathcal{U}_1, \mathcal{A}_1, \preceq_1, \leq_p^1 \rangle$ and $\mathbb{A}_2 = \langle \mathcal{L}_2, \mathcal{U}_2, \mathcal{A}_2, \preceq_2, \leq_p^2 \rangle$ are approximation frameworks such that $\boldsymbol{\mathcal{A}_1} \sim \boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{A}_2} \sim \boldsymbol{\mathcal{E}}$.

Definition 5.8.14. *Let $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$ and let $A_1 \in Appr(\boldsymbol{\mathcal{A}_1})$ and $A_2 \in Appr(\boldsymbol{\mathcal{A}_2})$.*

- $(A_1 \circ \iota)$ *is the* natural extension *of A_1 on $\boldsymbol{\mathcal{A}_2}$.*
- $(\iota \circ A_2)$ *is the* natural abstraction *of A_2 on $\boldsymbol{\mathcal{A}_1}$.*

The following properties summarize how the semantics associated with such pairs of operators relate. First let us compare an approximator A_1 on the less precise approximation space to its natural extension $(A_1 \circ \iota)$. Intuitively, the latter corresponds to the least informative extension of A_1 to $\boldsymbol{\mathcal{A}_2}$. For example let $\boldsymbol{\mathcal{A}_1}$ be the space of intervals while $\boldsymbol{\mathcal{A}_2}$ corresponds to flowers over a complete lattice. Then $(A_1 \circ \iota)$ maps any flower to the image of A_1 for the smallest interval containing this flower.

Proposition 5.8.15. *Let $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$. If $A_1 \in Appr(\boldsymbol{\mathcal{A}_1})$ and $A_2 = (A_1 \circ \iota)$, then*

- *$\boldsymbol{X}$ is a fixpoint of A_1 iff $\boldsymbol{X}$ is a fixpoint of A_2.*
- *$\boldsymbol{X}$ is a stable fixpoint of A_1 iff $\boldsymbol{X}$ is a stable fixpoint of A_2.*

Proof. Let $\boldsymbol{X}$ be a fixpoint of A_2, then $\boldsymbol{X} = A_2(\boldsymbol{X}) = A_1(\iota(\boldsymbol{X}))$. Hence, $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_1}$, on the other hand, let $\boldsymbol{X}$ be a fixpoint of A_1, then again $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_1}$. Thus in both cases, $\boldsymbol{X} = \iota(\boldsymbol{X})$ by Proposition 5.8.2. Then it follows that $A_2(\boldsymbol{X}) = A_1(\iota(\boldsymbol{X})) = A_1(\boldsymbol{X})$, thus $\boldsymbol{X} = A_1(\boldsymbol{X})$ iff $\boldsymbol{X} = A_2(\boldsymbol{X})$.

Let $\boldsymbol{X}$ be a stable fixpoint of A_1, then by Proposition 5.6.31, $\boldsymbol{X}$ is a fixpoint of A_1, thus $\boldsymbol{X}$ is a fixpoint of A_2 (and vice versa). By the reasoning above, we then again find that $\boldsymbol{X} = \iota(\boldsymbol{X})$. Clearly, if $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$ is a fixpoint of $(A_2)^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X^{\mathcal{U}}}}$, then $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} = (A_2)^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X^{\mathcal{U}}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}) = A_2^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}) = (A_1)^{\boldsymbol{\mathcal{L}}}(\iota(\boldsymbol{X})) = (A_1)^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}) = (A_1)^{\boldsymbol{\mathcal{L}}}_{\boldsymbol{X^{\mathcal{U}}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{L}}})$ and thus $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$ is a fixpoint

[2] Function composition is defined as usual, i.e., given functions $f : D_2 \to D_3$ and $g : D_1 \to D_2$, the function $(f \circ g) : D_1 \to D_3$ maps any $x \in D_1$ to $f(g(x))$.

of $(A_1)^{\mathcal{L}}_{\boldsymbol{X}^{\mathcal{U}}}$ and vice versa. Analogously, $\boldsymbol{X}^{\mathcal{U}}$ is a fixpoint of $(A_2)^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$ iff $\boldsymbol{X}^{\mathcal{U}}$ is a fixpoint of $(A_1)^{\mathcal{U}}_{\boldsymbol{X}^{\mathcal{L}}}$.

Since there is an embedding from $\boldsymbol{\mathcal{A}_1}$ to $\boldsymbol{\mathcal{A}_2}$, $\leq^2_p$ coincides with $\leq^1_p$ on $\boldsymbol{\mathcal{A}_1}$, hence the orders $\leq_1$ and $\leq_2$ will coincide on the sets of fixpoints. In other words, the least fixpoints are preserved by the embedding. Therefore $\boldsymbol{X}^{\mathcal{L}} = St^1_{A_1}$ iff $\boldsymbol{X}^{\mathcal{L}} = St^1_{A_2}$ and $\boldsymbol{X}^{\mathcal{U}} = St^2_{A_1}$ iff $\boldsymbol{X}^{\mathcal{U}} = St^2_{A_2}$. In conclusion, $\boldsymbol{X}$ is a stable fixpoint of A_1 iff $\boldsymbol{X}$ is a stable fixpoint of A_2. □

Proposition 5.8.16. *Let $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$. If $A_1 \in Appr(\boldsymbol{\mathcal{A}_1})$ and $A_2 = (A_1 \circ \iota)$, then*

- $KK(A_1) = KK(A_2)$.
- $WF(A_1) = WF(A_2)$.
- $SUP(A_1) = SUP(A_2)$.
- $ST(A_1) = ST(A_2)$.

Proof. The first two statements straightforwardly follows from Proposition 5.8.15 and the fact that there is an embedding from $\boldsymbol{\mathcal{A}_1} \to \boldsymbol{\mathcal{A}_2}$, preserving the precision order. Hence, the least (stable) fixpoints must coincide. The last two claims also follow from Proposition 5.8.15, this time combined with the fact that the embedding ψ preserves $\sim$, therefore, if a fixpoint $\boldsymbol{X}$ is exact in $\boldsymbol{\mathcal{A}_1}$, it must be exact in $\boldsymbol{\mathcal{A}_2}$, and vice versa. □

In conclusion, $(A_1 \circ \iota)$ performs exactly the same as A_1. As a result, we deduce that a more precise approximation space delivers at least the same level of expressiveness as the less precise approximation space. Notably, the approximator $(A_1 \circ \iota)$ essentially does not use any of the additional accuracy provided by $\boldsymbol{\mathcal{A}_2}$.

On the other hand, consider an approximator $A_2 \in Appr(\boldsymbol{\mathcal{A}_2})$ and its natural abstraction $(\iota \circ A_2)$. Intuitively, such derived operator approximates the behavior of A_2 on the less precise approximation space $\boldsymbol{\mathcal{A}_1}$. For example, if we again look at iAFT compared to fAFT, then any operator A_2 on the flowers leads to an operator on intervals by mapping an interval $\boldsymbol{X}$ to the smallest interval containing the flower $A_2(\boldsymbol{X})$. Naturally, such an operator may lose information, hence, we expect that, in general, the expressiveness of A_2 surpasses that of $(\iota \circ A_2)$.

Lemma 5.8.17. *If $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$ under $\iota : \boldsymbol{\mathcal{A}_2} \to \boldsymbol{\mathcal{A}_1}$, then for every $A_2 \in Appr(\boldsymbol{\mathcal{A}_2})$, $(\iota \circ A_2 \circ \iota) \leq_p A_2$.*

Proof. Clearly by Proposition 5.8.3, for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_2}$, $\iota(\boldsymbol{X}) \leq_p \boldsymbol{X}$, then by monotonicity of A_2, $A_2(\iota(\boldsymbol{X})) \leq_p A_2(\boldsymbol{X})$. Moreover, again by Proposition 5.8.3, $\iota(A_2(\iota(\boldsymbol{X}))) \leq_p A_2(\iota(\boldsymbol{X}))$. Hence, for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_2}$, $(\iota \circ A_2 \circ \iota)(\boldsymbol{X}) \leq_p A_2(\boldsymbol{X})$. As a result $(\iota \circ A_2 \circ \iota) \leq_p A_2$. □

Proposition 5.8.18. *Let $\mathcal{A}_1 \leq_p \mathcal{A}_2$ under $\iota : \mathcal{A}_2 \to \mathcal{A}_1$. If $A_2 \in Appr(\mathcal{A}_2)$ and $A_1 = (\iota \circ A_2)$, then*

- $KK(A_1) \leq_p KK(A_2)$.
- $WF(A_1) \leq_p WF(A_2)$.
- $SUP(A_1) \subseteq SUP(A_2)$.
- *if $\mathcal{A}_2$ is non-degenerate, then $ST(A_1) \subseteq ST(A_2)$.*

Proof. From Proposition 5.8.16, we know that $KK(A_1) = KK(A_1 \circ \iota) = KK(\iota \circ A_2 \circ \iota)$ and $WF(A_1) = WF(A_1 \circ \iota) = WF(\iota \circ A_2 \circ \iota)$. Then by Lemma 5.8.17 and Theorem 5.8.11 we obtain $KK(A_1) = KK(\iota \circ A_2 \circ \iota) \leq_p KK(A_2)$ and $WF(A_1) = WF(\iota \circ A_2 \circ \iota) \leq_p WF(A_2)$.

Similarly, from Proposition 5.8.16, we know that $SUP(A_1) = SUP(A_1 \circ \iota)$ and $ST(A_1) = ST(A_1 \circ \iota)$. Then by Lemma 5.8.17 and Theorem 5.8.12 we find $SUP(A_1) = SUP(A_1 \circ \iota) = SUP(\iota \circ A_2 \circ \iota) \subseteq SUP(A_2)$. Similarly, if $\mathcal{A}_2$ extends $\mathcal{E}$, then by Lemma 5.8.17 and Theorem 5.8.13, $ST(A_1) = ST(A_1 \circ \iota) = ST(\iota \circ A_2 \circ \iota) \subseteq ST(A_2)$. □

Proposition 5.8.19. *Let $\mathcal{A}_1 \leq_p \mathcal{A}_2$ under $\iota : \mathcal{A}_2 \to \mathcal{A}_1$. If O has an ultimate approximator U_2 for $\mathcal{A}_2$, then $(\iota \circ A_2)$ is a quasi-ultimate approximating operator of O for $\mathcal{A}_1$.*

Proof. Assume there exists an approximator $A_1 \in Appr(\mathcal{A}_1)$ that approximates O such that $(\iota \circ U_2) \leq_p A_1$. Then we have for every $\boldsymbol{X} \in \mathcal{A}_1$, $\iota(U_2(\boldsymbol{X})) = (\iota \circ U_2)(\boldsymbol{X}) \leq_p A_1(\boldsymbol{X})$. Moreover, since U_2 is the ultimate approximator of O in $\mathcal{A}_2$, for every other approximator $A_2 \in Appr(\mathcal{A}_2)$ of O, it must hold that $A_2 \leq_p U_2$. We know $(A_1 \circ \iota) \in Appr(\mathcal{A}_2)$, therefore $(A_1 \circ \iota) \leq_p U_2$. As a result, for every $\boldsymbol{X} \in \mathcal{A}_1$, $A_1(\boldsymbol{X}) = A_1(\iota(\boldsymbol{X})) = (A_1 \circ \iota)(\boldsymbol{X}) \leq_p^2 U_2(\boldsymbol{X})$, where the first equality follows from Proposition 5.8.2. Then by choice of ι, we find $A_1(\boldsymbol{X}) \leq_p^2 U_2(\boldsymbol{X})$ implies $A_1(\boldsymbol{X}) \leq_p^1 \iota(U_2(\boldsymbol{X}))$. In summary, we obtain $\iota(U_2(\boldsymbol{X})) = A_1(\boldsymbol{X})$. Since $\boldsymbol{X}$ was arbitrarily chosen, $A_1 = (\iota \circ U_2)$. Therefore, $(\iota \circ U_2)$ must be $\leq_p$-maximal, and thus quasi-ultimate. □

Example 5.8.20. *Let the exact space $\mathcal{E}$ be a complete lattice. We know that $\mathcal{A}_i(\mathcal{E}) \leq_p \mathcal{A}_f(\mathcal{E})$ through ι which maps any flower $\boldsymbol{X}$ to the interval $[\mathrm{glb}(\boldsymbol{X}), \mathrm{lub}(\boldsymbol{X})]$. Therefore, for any approximator A_i in iAFT, there is an approximator in fAFT with at least the same expressiveness, i.e., the natural extension of A_i on $\mathcal{A}_f(\mathcal{E})$. On the other hand, for any approximator A_f in fAFT,*

there is an approximator in iAFT with at most the same expressiveness, i.e., the natural abstraction of A_f on $\boldsymbol{\mathcal{A}_i}(\boldsymbol{\mathcal{E}})$.

Furthermore, the natural abstraction of the ultimate approximator of a semantic operator O in fAFT, i.e., U_f, is the ultimate approximator of O in iAFT, i.e., U_i, by Proposition 5.8.19.

Proposition 5.8.21. *Let $A_i \in Appr(\boldsymbol{\mathcal{A}_i}(\boldsymbol{\mathcal{E}}))$ and let $A_f \in Appr(\boldsymbol{\mathcal{A}_f}(\boldsymbol{\mathcal{E}}))$. If $WF(A_f)$ is exact and $x \in ST(A_i)$, then $WF(A_f) = \{x\}$.*

Proof. Clearly, if $A_i \in Appr(\boldsymbol{\mathcal{A}_i}(\boldsymbol{\mathcal{E}}))$, then $A_i \leq_p U_i$. Similarly, $A_f \leq_p U_f$. Therefore, by Theorem 5.8.11, $WF(A_f) \leq_p WF(U_f)$. Thus, since $WF(A_f)$ is exact, i.e., $\leq_p$-maximal, $WF(A_f) = WF(U_f)$. Similarly, by Theorem 5.8.13, $ST(A_i) \subseteq ST(U_i)$ since $\boldsymbol{\mathcal{A}_f}(\boldsymbol{\mathcal{E}})$ is conservative. Furthermore, by Proposition 5.8.18, also $ST(U_i) \subseteq ST(U_f)$. Consequently, if $x \in ST(A_i)$, then $x \in ST(U_f)$ and thus $WF(U_f) \sim x$ by Proposition 5.6.29. Hence $x \in WF(U_f)$. Thus, since by assumption $WF(U_f)$ is exact, we find $WF(A_f) = WF(U_f) = \{x\}$. □

In other words, if iAFT has a unique stable model and fAFT induces an exact well-founded model, then these correspond. Notably, this supports the idea that flowers, and in general any generalized approximation space, do not lead to 'new' semantics and 'new' notions of fixpoints, instead they improve and refine the construction of these notions.

In summary, as expected, more precise approximation spaces are capable of reaching higher accuracy when sufficiently precise approximators are considered. However, increasing the precision of the approximator generally also increases the computational complexity. Therefore, the choice of approximator, and of the associated approximation space, should strike a balance between expressiveness and complexity.

Note, however, that if $\boldsymbol{\mathcal{A}_1} \leq_p \boldsymbol{\mathcal{A}_2}$, $\boldsymbol{\mathcal{A}_1}$ is essentially a subspace of $\boldsymbol{\mathcal{A}_2}$. Now, let $A_1 \in Appr(\boldsymbol{\mathcal{A}_1})$ and $A_2 \in Appr(\boldsymbol{\mathcal{A}_2})$ such that $(A_1 \circ \iota) \leq_p A_2$. Then $KK(A_1) \in \boldsymbol{\mathcal{A}_2}$ and $KK(A_1) \leq_p KK(A_2)$. This entails that if we are not satisfied with the accuracy of A_1, we do not need to start from scratch when calculating the Kripke-Kleene fixpoint with A_2 in the more precise space. Instead, we can start from $KK(A_1)$. Similarly, we know that for every $\boldsymbol{X}_1 \in \boldsymbol{\mathcal{A}_1}$, if $\boldsymbol{X}_1$ is A_1-reliable and -prudent, then $\boldsymbol{X}_1 \in \boldsymbol{\mathcal{A}_2}$ is A_2-reliable and -prudent. Hence, also for the well-founded model a complete restart is unnecessary. Consequently, instead of using a fixed approximation space with a fixed approximator, we can refine the space and approximator ad hoc. Specifically, assume we have at our disposal a finite chain of approximation spaces and approximators, i.e., $\langle \boldsymbol{\mathcal{A}_k}, A_k \rangle_{k \leq n}$ such that for every $k \leq j \leq n$, $\boldsymbol{\mathcal{A}_k} \leq_p \boldsymbol{\mathcal{A}_j}$ and $(A_k \circ \iota_{j \to k}) \leq_p A_j$.

Algorithm 5.8.1 Iteratively Refined Computation of the Well-Founded Semantics

$k \leftarrow 0$
$\boldsymbol{X} \leftarrow \bot_{\boldsymbol{\mathcal{A}_0}}$
while $\boldsymbol{X}$ is not exact and $k \leq n$ **do**
  while $\boldsymbol{X} \neq St_{A_k}(\boldsymbol{X})$ **do**
    $\boldsymbol{X} \leftarrow St_{A_k}(\boldsymbol{X})$
  $k \leftarrow k + 1$

For example, let us consider Algorithm 5.8.1 to compute the well-founded semantics. We start with the least precise and least complex approximation space and approximator, i.e., $\boldsymbol{\mathcal{A}_0}$ and A_0. For this we compute the well-founded fixpoint by the fastest monotonic induction of the stable revision operator. As long as said fixpoint is not sufficiently accurate, e.g., it is not exact, we move on to the next, more precise approximation space and approximator and reprise the process from the previously derived fixpoint. Essentially, this iterative procedure implicitly constructs a well-balanced approximation space and approximator. Specifically, such combination of approximators can be rephrased as a single approximator that prefers less costly operations as long as those provide at least some additional knowledge. Consequently, the operator is generally more costly for more precise approximants. Note that, compared to the fastest monotonic induction of the stable revision operator St_{A_n}, this iterative procedure typically increases the number of iterations necessary to find the well-founded fixpoint while decreasing the average computational complexity of each step. In other words, this procedure is only beneficial if the decreased step-wise complexity more than compensates for the added steps. Fortunately, as we have seen for iAFT and fAFT, for plenty natural approximation spaces the number of iteration steps is bound by (or polynomial in) the size of the exact space, hence the increase in number of iterations will often be negligible compared to the efficiency gain per step. In summary, the hierarchy of approximation spaces and approximators allows us to dynamically balance precision and expressiveness with computational efficiency.

5.9 Conclusion

This chapter proposes a conservative extension of consistent (interval-)AFT by relaxing the constraints on the approximation space. With this extension the approximation space need not be intervals, more general approximants are allowed. This generalization covers more than the usual approximation spaces while ensuring the expected behavior. The work is motivated by examples from different non-monotonic reasoning logics where standard AFT fails. On the one hand, we see that standard AFT can only handle problems in which the operator operates on a complete lattice. However, in

many practical applications, such as for abstract argumentation frameworks or while constructing (co-)recursive definitions, this is not the case. Additionally, even when iAFT can be applied, it sometimes lacks expressive power, as was indicated by an example from auto-epistemic logic. By allowing more general approximants, AFT is capable of resolving these issues as will be demonstrated further in the following chapter. This suggests that the extension improves expressiveness. Nevertheless, it remains unclear how far those improvements go. In fact, one might argue that it is unclear if the theoretical expressiveness was improved at all. Perhaps any of our general approximation spaces can be extended to a cpo fitting in iAFT, that is at least as precise. However, arguably, the space of flowers of fAFT is more natural than any consistent subposet of a bi-lattice that is at least as precise as this space. Moreover, while it is common practice to extend the exact space to a complete lattice, such extensions are made ad hoc and they have never formalized or investigated. However, as demonstrated by Example 5.6.23, caution is warranted when extending an exact space.

Additionally, by zooming in on the requirements of AFT, the work gives new insights on the key properties of AFT. In fact, this proposal is arguably the most general possible conservative extension of consistent AFT provided that a conservative extension adheres to all requirements that were formulated in Section 5.3. It is important to remark that although this framework is the most general conservative extension of consistent AFT in the context of the proposed requirements, it does not cover every version of AFT. For example, at first sight, it does not include disjunctive AFT, nor have we investigated the relation between disjunctive AFT and this generalized AFT. Perhaps there might be an approximation space with approximator (and stable revision operator) that are isomorphic to those used in disjunctive AFT.

Further, the general framework allows us to compare semantics across different approximations spaces. Not only does this introduce additional flexibility, it also improves our understanding of AFT by underlining the core principles. However, by introducing a new level of flexibility in the approximation space, the extension also poses a new challenge. Since there are multiple possible approximation spaces, it is not necessarily immediately clear which approximation space is best suited for an application. A good approximation space for an application strikes a balance between expressive power and computational complexity. After examining the relations between approximation spaces, we see that we do not need to decide on a fixed approximation space from the beginning; instead we can iteratively refine our approximation space ad hoc, essentially resulting in a well-balanced approximation space.

We are not the first to pursue a generalization of ideas within the AFT framework. For instance, Charalambidis and Rondogiannis [21] introduced categorical AFT (cAFT), in which arbitrary categories can serve as exact spaces, thereby abstracting away from the traditional notion of an exact space. Their approach was motivated by examples where the semantic operator does not act on a complete lattice, such as in Examples 5.1.1 and 5.1.2. However, while our extension constitutes a conservative one, preserving

the key theorems and desirable properties of AFT, the same does not hold for cAFT. In particular, within cAFT the existence of the various types of fixpoints is no longer guaranteed. In other words, their framework trades certain theoretical guarantees for greater generality and flexibility. We believe that an interesting direction for future research lies in exploring how our generalized notion of approximation spaces might support alternative formulations of cAFT, where the approximation space depends on the underlying category in such a way that these desirable properties can be recovered.

Another generalizing effort was undertaken by Pollaci et al. [71, 72], who also approached AFT from within the domain of category theory. In their work, the authors provide a general method to lift existing versions of AFT to higher-order settings by assuming a Cartesian closed category of approximation spaces for which AFT is known to work. Their approach imposes relatively few restrictions on such category, specifically, that all approximation spaces must be cpos, but introduces stronger constraints on the exact approximants within those spaces. For instance, the set of exact approximants representing the same exact element must possess a greatest lower bound, a condition not necessarily met in our generalized framework. Nevertheless, we believe that many natural instances within our framework form categories that satisfy these constraints. This opens another promising avenue for future research: identifying which versions of our framework can be lifted to higher-order settings through this categorical perspective.

Finally, while we have proposed a framework for generalized *consistent* AFT, we have not yet investigated whether we can extend our framework to the more general idea of non-consistent AFT. It has been shown for iAFT that an approximation space and approximator can be extended to a complete lattice for which sAFT applies. For our general framework, we hypothesize that it is theoretically possible to transform approximation spaces and approximators in a similar fashion. However, we wonder whether these are sensible transformations. Already in sAFT it is hard to explain what knowledge an inconsistent element embodies, but this becomes even more difficult for inconsistent approximants in general. Moreover, while for sAFT, the elegance of the product space over the consistent subspace is a reasonable argument in favor of including the somewhat ‘unnatural’ inconsistent elements, the same cannot really be said about a non-consistent extension of general consistent approximation spaces. For example, in fAFT, we need careful consideration of the order, decomposition and recomposition functions if we want to extend it to a product space of the lower- and upper- decomposition spaces. In the end, due to the additional overhead associated with aforementioned functions and order, a non-consistent extension is hardly more elegant.

Chapter 6

Practical Applications of Generalized AFT and Flower-based AFT

6.1 Introduction

An abstract theory is only as valuable as its applications. Therefore, whereas the previous chapter focused on theoretical results, we conclude this work with a chapter devoted to practical applications of those results. First this chapter focuses on non-degenerate approximation spaces, a class for which the previous chapter established promising semantic properties, most notably the correctness property. Among these spaces lies fAFT, which serves as a central example. More specifically, we revisit all motivating examples and domains introduced in the previous chapters and demonstrate that fAFT already overcomes all the pitfalls we had identified for iAFT. While fAFT is certainly not the only instance of our framework, it is especially noteworthy because it offers significant improvements in expressiveness compared to iAFT, while still providing a natural, clear, and intuitive approximation space. Throughout Chapter 5 we have already worked out Example 5.1.3 showcasing the improved expressiveness of fAFT with respect to AEL theories. Here, Section 6.2 starts by elaborating further on this and discusses fAFT for AEL in comparison to some of the results formulated in the work by Bogaerts et al. [11].

Then, Section 6.3 focuses more on the usability of fAFT other non-degenerate approximation spaces, compared to iAFT. Whereas iAFT requires the exact space

$\mathcal{E}$ to be a complete lattice, other AFT- versions, such as fAFT, only require $\mathcal{E}$ to be a bounded-complete cpo. Plenty of problems constitute an operator on a bounded-complete cpo which is not a complete lattice, as shown by Examples 5.1.1 and 5.1.2. Arguably, a bounded-complete cpo can be easily extended to a complete lattice by adding the relevant top-elements. Accordingly, we also study a non-degenerate iAFT-like approximation space that is defined over this enriched exact space. Nevertheless, fAFT provides a more natural setting in which the behavior of operators can be studied directly on the underlying space. This makes fAFT not only more flexible, but also more faithful to the original problem structure, avoiding the need for artificial extensions that may obscure the semantics of the studied problems. Moreover, we show that, the simplest and most natural of such extensions do not offer the benefits typically associated with AFT. In particular, generally they do not allow us to keep track of sufficiently refined upper bounds during construction of the least fixpoint for monotonic operators, nor do they suffice to find the solution for non-monotonic operators. In fact, we notice that for the latter, even fAFT mostly fails, suggesting we need a more precise instance of our generalized framework.

The next section examines how the work on AFT in a context of normal logic programming can be extended to include fAFT. In particular, whereas the ultimate operator for fAFT is well-defined, it is typically not the most appropriate approximator for logic programs. For example, Chapter 4 discussed Fitting's three-valued immediate consequence operator and extensions of it based on a three-valued truth assignment. Such approximators seemingly better match the point of view and intuitions of the ASP and LP-community. For this reason, Section 6.4 constructs a family of fAFT-approximators which make use of the improved expressiveness of the space, while also presenting some of the desirable properties of Fitting's three-valued immediate consequence operator.

In the last part, we turn to an application that employs an approximation space which is not non-degenerate. Specifically, we revisit a variant of Example 5.1.3 approached as a logic program. Interestingly, by allowing more flexibility with respect to the approximation space of an exact space, we are now able to play around with different approximation spaces typically associated with other non-monotonic reasoning logics. For example, here we show that any approximation space of the exact space associated with the logic program we proposed as an alternative modeling of Example 5.1.3, is also an approximation space of the exact space of belief sets, and similarly for the approximator. Interestingly, despite the fact that the approximation space is not non-degenerate, the semantics continue to behave in line with our expectations. This observation suggests that the non-degeneracy requirement may be too restrictive, and it motivates future work aimed at relaxing the conditions under which we can guarantee the correctness of the semantics. Then, examples like this suggest that the generalized AFT (gAFT) could provide us with tools to formally compare different semantics for a given problem going beyond just one non-monotonic reasoning logic.

While this work establishes several theoretical and practical contributions of gAFT, many questions remain open. The full scope of its practical implications, particularly in complex domains, is yet to be explored. This chapter aims to illustrate key possibilities and provide a foundation for future investigation.

6.2 fAFT and Expressiveness in Auto-Epistemic Logic

At first glance, flowers positively contribute to the expressiveness of AFT for AEL, as demonstrated by Example 5.1.3. Interestingly, although iAFT fails to solve it, this example is a fairly simple auto-epistemic theory. In fact, despite its non-monotonic character, it is relatively well-behaved. Specifically, Bogaerts et al. [11] categorize this example as a *strict monotonically stratifiable* AEL-theory. Intuitively, in such theories, we can assign strata or levels to the different atoms in our alphabet Σ such that our objective knowledge about an atom $s \in \Sigma$ can only depend on (1) the objective knowledge about atoms t such that the stratum of t is at most equal to that of s, or (2) the modal knowledge about atoms t such that the stratum of t is strictly smaller than that of s. Additionally, they identify a strictly larger category of theories, i.e., the *monotonically stratifiable* AEL-theories, where the objective knowledge about s may also negatively depend on modal formulas $K\psi$ for which all atoms that occur in ψ live in the same stratum as s. For a more precise and formal definition of these categories, we refer the reader to Definition 3.2 by Bogaerts et al. [11].

Interestingly, for strict monotonically stratifiable theories, the semantic operator has always only one fixpoint and iteratively applying the operator will find the intended fixpoint, independent of the belief state we start from. Because of this, the Kripke-Kleene semantics proposed by Bogaerts et al. [11] is strong enough to find the intended fixpoint for all these theories. Similarly, as exemplified by Example 5.1.3, we expect the same to be true for the ultimate approximator in fAFT.

Conjecture 6.2.1. *Let $\mathcal{T}$ be a strict monotonically stratifiable theory, and let $O_{\mathcal{T}}$ be the semantic operator associated with $\mathcal{T}$. If U_f is the ultimate approximator for $O_{\mathcal{T}}$ in fAFT, then $KK(U_f) = \{\mathrm{lfp}(O_{\mathcal{T}})\}$.*

In contrast, while Bogaerts et al. [11] also provides well-founded semantics that identify the correct intended fixpoint for monotonically stratifiable theories, we cannot claim the same, as demonstrated by the following example.

Example 6.2.2. *Consider the following slight extension of the AEL-theory in Example 5.1.3:*

$$\mathcal{T} = \left\{ \begin{array}{c} q \Leftrightarrow \neg K p. \\ r \Leftrightarrow \neg K q. \\ r \vee \neg K r. \end{array} \right\}.$$

Interestingly, this extended example only adds a natural rule that entails that if we know that r holds, then r must hold. While, at first sight, this extension is innocent enough, the theory is no longer strictly monotonically stratifiable. Specifically, we see that r depends on $\neg K r$. Nevertheless it is still (non-strictly) monotonically stratifiable.

Crucially, the semantic operator associated with this example, now maps the belief state where q and r are known to the inconsistent belief state. Let us now look at the ultimate semantics for this operator:

Iteration 1

$$\begin{aligned} U_f^{\mathcal{L}}([\![2^\Sigma, \{\emptyset\}]\!]) &= [\![2^\Sigma, \{\emptyset\}]\!]^{\mathcal{L}} \\ &= \bot \end{aligned}$$

As a result, $St^1_{U_f}([\![2^\Sigma, \{\emptyset\}]\!]) = 2^\Sigma$.

$$\begin{aligned} U_f^{\mathcal{U}}([\![2^\Sigma, \{2^\Sigma\}]\!]) &= \{\{q \wedge r\}^\Sigma\}^{\mathcal{U}} = \{\{q \wedge r\}^\Sigma\} = u_1 \\ &\downarrow \\ U_f^{\mathcal{U}}([\![2^\Sigma, u_1]\!]) &= [\![\{q\}^\Sigma, \{\emptyset\}]\!]^{\mathcal{U}} \\ &= \{\emptyset\} = u_2 \\ &\downarrow \\ U_f^{\mathcal{U}}([\![2^\Sigma, \{\emptyset\}]\!]) &= [\![2^\Sigma, \{\emptyset\}]\!]^{\mathcal{U}} = u_2 \end{aligned}$$

Therefore, $St^2_{U_f}([\![2^\Sigma, \{\emptyset\}]\!]) = u_2 = \{\emptyset\}$. *As a result,* $St_{U_f}([\![2^\Sigma, \{\emptyset\}]\!]) = [\![2^\Sigma, \{\emptyset\}]\!] = WF(U_f)$.

Hence, despite the increased expressiveness, fAFT still is not refined enough to solve all monotonically stratifiable theories. However, note that the lack of expressiveness for the previous example is a direct consequence of the fact that $O_{\mathcal{T}}$ is not consistency-preserving, i.e., although the belief state where q and r are known is a consistent belief state, it is mapped to the inconsistent belief state by $O_{\mathcal{T}}$. Because of this, we cannot refine the upper bound below the inconsistent belief state. We believe that if the semantic operator is consistency-preserving, fAFT will match the power of the well-founded sets as introduced by Bogaerts et al. [11].

Conjecture 6.2.3. *Let $\mathcal{T}$ be a monotonically stratifiable theory, and let $O_{\mathcal{T}}$ be the*

semantic operator associated with $\mathcal{T}$. If U_f is the ultimate approximator for $O_\mathcal{T}$ in fAFT and $O_\mathcal{T}$ is consistency-preserving, then $WF(U_f)$ finds the intended fixpoint of $O_\mathcal{T}$.

Nevertheless, there are plenty of natural auto-epistemic theories that are not consistency-preserving. However, arguably, a good, rational agent does not consider the inconsistent belief state to be possible. Therefore, for a rational agent, any semantic operator that is not consistency-preserving should be treated as a partial operator instead. As a result, instead of a normal (total) ultimate approximator, we could build a partial 'ultimate' approximator that maps any flower $\boldsymbol{X}$ of belief states to the smallest flower $\boldsymbol{Y}$ such that for every belief state $\mathcal{X} \in \boldsymbol{X}$, if $O_\mathcal{T}(\mathcal{X})$ is not inconsistent, then $O_\mathcal{T}(\mathcal{X}) \in \boldsymbol{Y}$. Assuming that St is defined completely analogously for partial approximators, we see that this approximator does find the intended fixpoint, following reasoning mirroring the derivation of the ultimate well-founded fixpoint of the original theory in Example 5.1.3. Therefore, while at first glance, Example 6.2.2 suggests that the expressiveness of the well-founded sets by Bogaerts et al. [11] is strictly better than that of fAFT, we instead hypothesize that the well-founded semantics of fAFT is sufficiently strong to solve all monotonically stratifiable theories, provided that we allow for partial approximators that ignore the derivation of the inconsistent belief state.

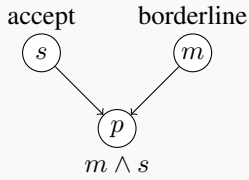
Conjecture 6.2.4. *Let $\mathcal{T}$ be a monotonically stratifiable theory, and let $O_\mathcal{T}$ be the semantic operator associated with $\mathcal{T}$. If U_p is the possibly partial ultimate approximator in fAFT associated with $O_\mathcal{T}$ where we ignore the derivation of the inconsistent belief state, then $WF(U_p)$ finds the intended fixpoint of $O_\mathcal{T}$.*

6.3 Improved Flexibility with Respect to the Exact Space

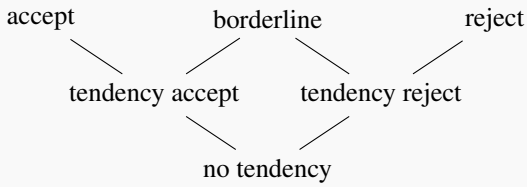
Throughout this work, we have shown that more precise approximation spaces like that of flowers $\mathcal{F}$ increase the expressiveness of AFT. Additionally, by generalizing the notion of an approximation space, we gain the flexibility to choose alternative approximation spaces that impose fewer restrictions on the exact space. In particular, while iAFT was defined only for exact spaces that form complete lattices, many other approximation spaces are more permissive. For example, fAFT only requires the exact space to be a bounded-complete cpo. As illustrated by Examples 5.1.1 and 5.1.2, many natural applications lead to exact spaces that are bounded-complete cpos but not complete lattices. This section explores problems where iAFT can not be employed straightforwardly and offers non-degenerate alternatives that fit in the general framework of gAFT.

6.3.1 Flower-AFT on Bounded Complete Cpos

Example 6.3.1 (Example 5.1.2, contd.)**.** *Recall the wADF from Example 5.1.2, which links the acceptance status of a paper* (p) *to its significance* (s) *and the used scientific methodology* (m)*. The dependency is represented by the acceptance-function* $m \wedge s$*. Moreover, let us again assume that the significance of the paper has the value 'accept'. On the other hand, peer reviewing determined that the scientific methodology should be assigned the value 'borderline'.*



(a) A simple wADF illustrating how the status of a paper depends on its significance and the used methodology.



(b) The acceptance values under $\leq_a$.

Generally, a wADF over a poset $\boldsymbol{V}$ of acceptance values can be characterized by a set S of *arguments* and a set $\Phi = \{\varphi_s\}_{s \in S}$ of *acceptance functions* $\varphi_s : (S \to \boldsymbol{V}) \to \boldsymbol{V}$. Intuitively, an acceptance function takes an assignment of acceptance values to all arguments and produces an acceptance value for one argument. We will accordingly denote a wADF by a tuple $\langle S, \Phi \rangle$. For the example above, S corresponds to $\{p, s, m\}$. Moreover Φ is the set of acceptance functions $\{\varphi_p, \varphi_s, \varphi_m\}$ such that for every $f \in (S \to \boldsymbol{V})$, $\varphi_p(f) := f(m) \wedge f(s)$, $\varphi_s(f) :=$ *accept* and $\varphi_m(f) :=$ *borderline*. We adopt the choices for the exact space and semantic operator from Bogaerts [10]. Interestingly, these notions are well-defined for any poset of acceptance values, irregardless of whether they form a complete lattice. Hence, they trivially generalize to bounded-complete cpos like the one considered in Example 6.3.1.

Definition 6.3.2. *Let* $\boldsymbol{V} = \langle V, \leq_a \rangle$ *be a poset of acceptance values and let* $\Xi = \langle S, \Phi \rangle$ *be a wADF. The exact space* $\boldsymbol{\mathcal{E}}$ *of* Ξ *for* $\boldsymbol{V}$ *is defined as the poset* $\langle \mathcal{E}, \leq_{pt} \rangle$ *where*

- *$\mathcal{E}$ is the set of functions with signature $S \to V$, and*
- *$\leq_{pt}$ is the point-wise extension of $\leq_a$ to $\mathcal{E}$.*

Definition 6.3.3. *Let $\boldsymbol{V} = \langle V, \leq_a \rangle$ be a poset of acceptance values and let $\Xi = \langle S, \{\varphi_s\}_{s \in S} \rangle$ be a wADF. The semantic operator $O : \boldsymbol{\mathcal{E}} \to \boldsymbol{\mathcal{E}}$ of Ξ is defined as the operator that maps an assignment $f \colon S \to V$ to*

$$O(f) := \Big\{ \forall s \in S : s \mapsto \varphi_s(f).$$

Example 6.3.4 (Example 6.3.1, contd.)**.** *In the recurring example the exact space $\boldsymbol{\mathcal{E}_{acc}}$ is the set of all functions f that assign acceptance values to the paper, its significance and the methodology, i.e., $f \colon \{p, s, m\} \to V$ ordered by the point-wise extension of $\leq_a$. Clearly, $\boldsymbol{\mathcal{E}_{acc}}$ has no greatest element. For example, consider the functions*

$$\begin{cases} p \mapsto \textit{accept} \\ s \mapsto \textit{accept} \\ m \mapsto \textit{accept} \end{cases} \quad \textit{and} \quad \begin{cases} p \mapsto \textit{borderline} \\ s \mapsto \textit{borderline} \\ m \mapsto \textit{borderline} \end{cases}$$

There is no function $f \in \boldsymbol{\mathcal{E}_{acc}}$ that is bigger than both. In fact both of these functions are maximal.

The semantic operator O_{acc} is defined to map an assignment $f \in \boldsymbol{\mathcal{E}_{acc}}$ to

$$O_{acc}(f) := \begin{cases} p \mapsto f(m) \wedge f(s) \\ s \mapsto \textit{accept} \\ m \mapsto \textit{borderline} \end{cases}$$

Interestingly, since the order on the exact space for a wADF is defined point-wise with respect to the order on the acceptance values, the structure of such exact space tends to be similar to that of the acceptance values $\boldsymbol{V}$. For example, if $\boldsymbol{V}$ is a complete lattice, then any wADF will produce a complete lattice for the exact space. This is why Bogaerts [10] only considers cases where $\boldsymbol{V}$ is a complete lattice, ensuring that the exact space is a complete lattice as well, as is essential for iAFT. By comparison, fAFT is less restrictive, i.e., it only requires the exact space to be a bounded-complete cpo. Crucially, wADFs as introduced by Brewka et al. [14] only allow posets of acceptance values $\boldsymbol{V}$ that form bounded-complete cpos.

Proposition 6.3.5. *Let $\boldsymbol{V}$ be a poset of acceptance values and let $\Xi = \langle S, \Phi \rangle$ be a wADF over $\boldsymbol{V}$. Let $\boldsymbol{\mathcal{E}}$ be the exact space of Ξ for $\boldsymbol{V}$. If $\boldsymbol{V}$ is a bounded-complete cpo, then $\boldsymbol{\mathcal{E}}$ is a bounded-complete cpo.*

Proof. Let $C \subseteq \boldsymbol{\mathcal{E}}$ be a chain. Since the order $\leq_{pt}$ is defined point-wise, for all $x \in S$, $C_x := \{f(x) \mid f \in C\} \subseteq \boldsymbol{V}$ is a chain. We know that $\boldsymbol{V}$ is a bounded-complete cpo. Consequently, for all $x \in S$, there exists $\mathrm{lub}_{\leq_a}(C_x)$. It is easy to see that the function $F\colon S \to \boldsymbol{V}$ defined such that $\forall x \in S : F(x) := \mathrm{lub}_{\leq_a}(C_x)$ is the least upper bound of C.

Let $S \subseteq \boldsymbol{\mathcal{E}}$ be a non-empty set. For all $x \in S$, $S'_x := \{f(x) \mid f \in S'\} \subseteq \boldsymbol{V}$ is a non-empty set as well. Clearly, since $\boldsymbol{V}$ is a bounded-complete cpo, for all $x \in V$, there exists $\mathrm{glb}_{\leq_a}(S'_x)$. It is easy to see that the function $F\colon S \to \boldsymbol{V}$ defined such that $\forall x \in S : F(x) := \mathrm{glb}_{\leq_a}(S'_x)$ is the greatest lower bound of S'. □

In other words, the exact space for this example is a bounded-complete cpo, but it is not a complete lattice as it does not have a greatest element. Consequently, we cannot apply iAFT for this wADF. In contrast, fAFT *is* applicable. Accordingly, the results expressed by Bogaerts [10] trivially generalize to all wADFs.

Example 6.3.6 (Example 6.3.4, contd.)**.** *Let us now consider the Kripke-Kleene fixpoint using the ultimate approximator of O_{acc} in fAFT, i.e., U_f. Once again, we derive the Kripke-Kleene fixpoint using the fastest monotonic induction of U_f:*

$$U_f(\boldsymbol{\mathcal{E}_{acc}}) = \{f \in \boldsymbol{\mathcal{E}_{acc}} \mid f(m) = \mathit{borderline} \wedge f(s) = \mathit{accept}\} = \boldsymbol{X}_1$$

$$\downarrow$$

$$U_f(\boldsymbol{X}_1) = \left\{ \left\{ \begin{array}{l} p \mapsto \mathit{tendency\ accept} \\ s \mapsto \mathit{accept} \\ m \mapsto \mathit{borderline} \end{array} \right\} \right\}$$

In summary, the fastest monotonic induction of U_f already finds the exact intended fixpoint after two iterations. We can consequently conclude that the well-founded semantics will also derive this fixpoint, i.e.,

$$KK(U_f) = \boldsymbol{X}_1 = WF(U_f).$$

An attentive reader might notice the strong resemblance between the definitions of on the one hand, the exact space for wADFs, and on the other hand, the construction space of (co-)recursive function definitions. In particular, to construct the construction space for a recursive definition Δ of a (total) function with signature $D \to B$, we assume a

definedness-order $\leq$ on $B \cup \{\bot\}$ where for every $b_1, b_2 \in B \cup \{\bot\}$, $b_1 \leq b_2$ iff $b_1 = \bot$ or $b_1 = b_2$. The construction space $\mathcal{C}$ is then the set of partial functions $D \to B \cup \{\bot\}$ ordered by the point-wise extension $\leq_d$ of the definedness-order $\leq$. In other words, if we take the set of arguments $S = D$, the set of acceptance values $V = B \cup \{\bot\}$ ordered by and acceptance order $\leq_a$ corresponding to the definedness-order $\leq$ on $B \cup \{\bot\}$, then independent of our choice for the acceptance functions Φ, the exact space $\mathcal{E}$ of the wADF $\langle S, \Phi \rangle$ for V is exactly the construction space $\mathcal{C}$ of Δ. Clearly, unless B is a singleton $\{b\}$, the poset $\langle B \cup \{\bot\}, \leq \rangle$ is a bounded-complete cpo, but not a complete lattice. Consequently, the construction space is not a complete lattice, but by Proposition 6.3.5 it must be a bounded-complete cpo. As a result, again fAFT is applicable to the construction space of (co-)recursive definitions, while iAFT is not.

6.3.2 Interval-based AFT on Bounded Complete Cpos

Traditionally, before we generalized the definition of an approximation space, applying AFT in these types of examples required extending the exact space into a complete lattice, along with an extension of the semantic operator. These extensions were typically constructed in an ad hoc manner, relying on a few natural, intuitive candidates. However, this approach assumed, without formal guarantees, that these extensions adequately preserved the behavior of the original operator and space. However, not all extensions are appropriate as exemplified by Example 5.6.23. With the generalized framework, we now have precise criteria that guarantee a form of correctness, i.e., the derived approximation space must be a non-degenerate approximation space of the original construction space.

One simple way to make 'interval-style' approximants is by extending any bounded-complete cpo of acceptance values $\boldsymbol{V}$ to a complete lattice, by adding a top-element $\top$. Then, by Proposition 2.1.37, this poset is a complete lattice. We denote this complete lattice by $\boldsymbol{V}^\top$. Accordingly, the associated exact space $\boldsymbol{\mathcal{E}}^\top$ of any wADF Ξ is then also a complete lattice. It is easily verified that the exact space $\boldsymbol{\mathcal{E}}$ of Ξ for $\boldsymbol{V}$ is an induced subposet of $\boldsymbol{\mathcal{E}}^\top$. With this, we can define a non-degenerate variant of iAFT[1] for bounded complete cpos.

Definition 6.3.7. *Let $\boldsymbol{\mathcal{E}}$ be a bounded complete cpo, and let $\boldsymbol{\mathcal{E}}^\top$ be the complete lattice obtained by adding a top-element to the acceptance values. We define:*

$$\boldsymbol{\mathcal{L}}_{i^*} = \boldsymbol{\mathcal{E}}$$

$$\boldsymbol{\mathcal{U}}_{i^*} = \boldsymbol{\mathcal{E}}^\top$$

$$\mathcal{A}_{i^*} = \{(l, u) \in \boldsymbol{\mathcal{E}} \times \boldsymbol{\mathcal{E}}^\top \mid l \leq_{pt} u\}$$

[1] Note that simply applying iAFT to $\boldsymbol{\mathcal{E}}^\top$ would not result in an non-degenerate approximation space.

$$\forall (l_1, u_1), (l_2, u_2) \in \mathcal{A}_{i^*} : (l_1, u_1) \leq_p (l_2, u_2) \textit{ iff } l_1 \leq_{pt} l_2 \wedge u_2 \leq_{pt} u_1$$

$$\forall (l, u) \in \mathcal{A}_{i^*} : [\![l, u]\!] = (l, u)$$

$$\forall (l, u) \in \mathcal{A}_{i^*} : (l, u)^{\mathcal{L}} = (l, u)^1 = l$$

$$\forall (l, u) \in \mathcal{A}_{i^*} : (l, u)^{\mathcal{U}} = (l, u)^2 = u$$

It is easy to see that $\boldsymbol{\mathcal{A}_{i^*}} = \langle \mathcal{A}, \leq_p \rangle$ is a composition poset of $\langle \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{E}}^{\top}, \leq \rangle$. Furthermore, as $\boldsymbol{\mathcal{E}}$ is an induced subposet of $\boldsymbol{\mathcal{E}}^{\top}$, also the Chain Interlattice Lub Property and the Interlattice Lb Property trivially hold. Therefore $\boldsymbol{\mathcal{A}_{i^*}}$ is an approximation space. Intuitively, this approximation space is very similar to that of iAFT on the extended cpo, except that here we have removed all intervals from the approximation space that are empty with respect to the original cpo.

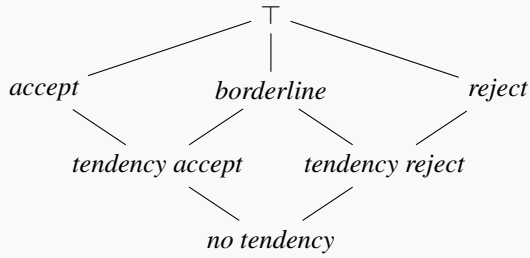
Proposition 6.3.8. *The poset $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$ is a non-degenerate extension of $\boldsymbol{\mathcal{E}}$.*

Proof. Clearly, as $\boldsymbol{\mathcal{L}_{i^*}} = \boldsymbol{\mathcal{E}}$, $\boldsymbol{\mathcal{A}_{i^*}}$ extends $\boldsymbol{\mathcal{E}}$. This entails that $\boldsymbol{\mathcal{A}_{i^*}}$ approximates $\boldsymbol{\mathcal{E}}$. We only need to show that for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}_{i^*}}$, $\boldsymbol{X}^{\boldsymbol{\mathcal{E}}\downarrow} = \boldsymbol{X}^{\boldsymbol{\mathcal{U}}\downarrow}$ is closed under the limits of chains. Naturally, for every chain C in $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}\downarrow}$, as $\leq_{pt}$ is the point-wise extension of $\leq_a$,

$$\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C) = \{\forall s \in S : s \mapsto \mathrm{lub}_{\boldsymbol{V}}(\{f(s) \mid f \in C\})$$

Furthermore, as $C \in \boldsymbol{X}^{\boldsymbol{\mathcal{U}}\downarrow}$, for every $f \in C$ and $s \in S$: $f(s) \leq_a \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}(s)$. Therefore $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}(s)$ must be an upper bound of $\{f(s) \mid f \in C\}$, hence $\mathrm{lub}_{\boldsymbol{V}}(\{f(s) \mid f \in C\}) \leq_a \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}(s)$. Thus, $\mathrm{lub}_{\boldsymbol{\mathcal{E}}}(C) \leq_{pt} \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. □

Example 6.3.9 (Example 6.3.1, contd.)**.** *For the simple example of paper acceptance, the extended poset of acceptance values is then characterized by the following Hasse diagram:*



If U_{i^*} is the ultimate approximator of O_{acc} on $\boldsymbol{\mathcal{A}_{i^*}}$, then the fastest monotonic induction of U_{i^*} is given by:

$$U_{i^*}(\bot_{\boldsymbol{\mathcal{A}_{i^*}}}) = \left(\begin{cases} p \mapsto \textit{no tendency} \\ s \mapsto \textit{accept} \\ m \mapsto \textit{borderline} \end{cases}, \begin{cases} p \mapsto \top \\ s \mapsto \textit{accept} \\ m \mapsto \textit{borderline} \end{cases} \right) = (g_1, h_1)$$

$$\downarrow$$

$$U_{i^*}(g_1, h_1) = \left(\begin{cases} p \mapsto \textit{tendency accept} \\ s \mapsto \textit{accept} \\ m \mapsto \textit{borderline} \end{cases}, \begin{cases} p \mapsto \textit{tendency accept} \\ s \mapsto \textit{accept} \\ m \mapsto \textit{borderline} \end{cases} \right)$$

Similar to U_f, the fastest monotonic induction of U_{i^*} finds the exact intended fixpoint after two iterations.

Hence, with only minimal overhead, we are able to identify an interval-based approximation space for our example which successfully finds the intended fixpoint. This raises the question of whether fAFT 's relaxed requirements on the exact space, compared to iAFT, are practically meaningful or simply theoretical and easily bypassed.

6.3.3 AFT on Monotonic Operators

As shown by Example 6.3.9, for some simple wADFs, it suffices to apply a relaxed version of iAFT. Specifically, if the wADF has no non-monotonic cycles in the dependency graph, then relaxed iAFT will be just as expressive as fAFT with respect to the well-founded fixpoint. This is to be expected as in such examples the corresponding semantic operator is monotonic, hence the upper bound or *AUB* is non-essential in finding the well-founded fixpoint. Nevertheless, even for monotonic operators, information about an upper bound can be significant. To illustrate this, consider the following simple recursive definition.

Example 6.3.10. *The function $Dbl\colon \mathbb{N} \to \mathbb{N}$, mapping each natural number n to double its value, can be defined recursively as follows:*

$$\left\lfloor \begin{array}{l} \mathit{Dbl}(0) := 0. \\ \forall n \in \mathbb{N} : \mathit{Dbl}(n+1) := \mathit{Dbl}(n) + 2. \end{array} \right\rfloor$$

If we interpret this within the context established in Chapter 3, the object space

$\mathcal{O}_{Dbl}$ is now the set of all functions with signature $\mathbb{N} \to \mathbb{N}$. Accordingly, given the observations on recursive definitions in Chapter 3, the construction space is the set of partial *functions $\mathbb{N} \nrightarrow \mathbb{N}$ equipped with the definedness-order, i.e., $\mathcal{C}_{Dbl} = \langle (\mathbb{N}_\perp)^{\mathbb{N}}, \leq_d \rangle$ where $\leq_d$ is the point-wise extension of the definedness order[2]$\leq$ on $\mathbb{N}_\perp$.*

Analogously to the examples of recursive definitions in Chapter 3, the semantic operator corresponds to rule application. In other words, the semantic operator $O_{Dbl} \colon \mathcal{C}_{Dbl} \to \mathcal{C}_{Dbl}$ is defined to map a function $f \in \mathcal{C}_{Dbl}$ to

$$O_{Dbl}(f) := \begin{cases} 0 & \mapsto 0 \\ x+1 & \mapsto f(x)+2 \end{cases}$$

It is not hard to see that O_{Dbl} is a monotonic operator on a bounded-complete cpo. As a result, any terminal monotonic induction of O_{Dbl} will obtain the intended fixpoint. Note however, that such a monotonic induction is infinitely long. Practically, instead of constructing the exact defined object in its entirety, the focus is typically on specific properties of interest. For instance, one may only need the mapping of a particular domain element, or aim to understand general tendencies rather than compute the whole fixpoint. For example, suppose we want to know whether Dbl can map a natural number to an odd number. Using monotonic inductions of O_{Dbl}, this question could only be answered once the fixpoint is reached, after infinitely many steps. By contrast, AFT and well-founded inductions allow us to exploit upper bounds to answer such questions without having to compute the entire defined object. In particular, grounding refinements can be used to tighten the upper bound so that the approximant only accounts for functions that map to even numbers.

Example 6.3.11 (Example 6.3.10, contd.)**.** *Suppose we want to know whether Dbl can map a natural number to an odd number. As we can treat recursive definitions as wADFs with elaborate acceptance functions, Proposition 6.3.5 implies that $\mathcal{E}_{Dbl} = \mathcal{C}_{Dbl}$ is a bounded-complete cpo. As a result, we can apply fAFT. Take $\boldsymbol{X} \in \mathcal{A}_f(\mathcal{E}_{Dbl})$ such that*

$$\boldsymbol{X} = \{f \in \mathcal{E}_{Dbl} \mid \forall n \in \mathbb{N} : f(n) = \perp \vee \mathit{Even}(f(n))\}$$

It is not hard to see that $\boldsymbol{X}$ must be a flower with $\mathrm{glb}(\boldsymbol{X}) = f_\perp$, i.e., the fully undefined function which maps every natural number to $\perp$. In other words, if $\boldsymbol{X} \leq_p A(\boldsymbol{X})$, then $\boldsymbol{X}$ is a grounding refinement of $\mathcal{E}_{Dbl}$. For example, using the

[2]Confer Chapter 3, the definedness order $\leq$ on $\mathbb{N}_\perp$ refers to the order such for every $x, y \in \mathbb{N}_\perp$, $x \leq y$ iff $x = y$ or $x = \perp$.

ultimate approximator U_f of O_{Dbl}, we obtain

$$U_f(\boldsymbol{X}) = \left\{ f \in \boldsymbol{\mathcal{E}_{Dbl}} \;\middle|\; \begin{array}{l} f(0) = 0 \wedge \\ \forall n \in \mathbb{N}\backslash\{0\} : f(n) = \bot \vee (Even(f(n)) \wedge f(n) \geq 2) \end{array} \right\}$$

Therefore, $\boldsymbol{X} \supseteq U_f(\boldsymbol{X})$ and $\boldsymbol{X}$ is indeed a grounding refinement of $\mathcal{E}_{Dbl}$. As a result, we can safely conclude that Dbl can never map any value to an odd number without ever computing the defined function in its entirety.

Similarly as for wADFs, we can extend the exact space, i.e., the construction space, of recursive definitions into a complete lattice by including a greatest 'most defined' element $\top$ in the co-domain. Then $\boldsymbol{\mathcal{E}_{Dbl}}^{\top}$ is a complete lattice with least element $f_\bot$ and greatest element $f_\top$ respectively mapping everything to $\bot$ and $\top$. However, whereas fAFT is a natural fit for $\boldsymbol{\mathcal{E}_{Dbl}}$ and effortlessly offers viable *AUBs* to solve questions of this kind, the same cannot be said of relaxed iAFT with $\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{E}_{Dbl}}^{\top}$. Specifically, it lacks the precision to express fine-grained information about upper bounds. They offer no means to constrain the possible values of a domain element unless it has at most one possible value remaining. As a result, in this approximation space, it is impossible to precisely represent the set of all functions that map every natural number to at most an even number. Consequently, this set cannot be used as a grounding refinement. In fact, a pair $(f_\bot, h)$ can only be a grounding refinement of $(f_\bot, f_\top)$ if $Dbl \leq_d h$. In other words, the only candidate grounding refinement precise enough to exclude functions that map a natural number to an odd number is $(f_\bot, Dbl)$. In summary, solving our problem essentially requires computing or knowing the entire defined object, analogous to monotonic inductions of O_{Dbl}. Thus, for monotonic operators on a bounded-complete cpo, fAFT yields tangible practical benefits in comparison to applying relaxed iAFT.

6.3.4 AFT on Non-Monotonic Operators

Of course, one of the main contributions of AFT is its ability to deal with non-monotonic operators. As discussed earlier, within the framework of constructive definitions, AFT has been shown to provide suitable semantics for non-monotonic inductive definitions. While iAFT falls short when it comes to capturing the exact space needed for (co-)recursive definitions, fAFT appears to be a natural fit. Indeed, revisiting Example 5.1.1, we see that fAFT provides the intended solution.

Example 6.3.12 (Example 5.1.1, contd.)**.** *Let $\mathcal{G} = (V, E)$ be a directed graph with vertices V and edges E. The distance between two vertices x and y is defined as the length of the shortest path between them, if it exists. One can define this*

distance function $d : V \times V \to \mathbb{N}_\perp$ formally as follows:

$$\left\lfloor \begin{array}{l} \forall x \in V : \quad d(x,x) := 0. \\ \forall x,z \in V : d(x,z) := \mathit{Min}(S) + 1 \leftarrow S = \left\{ d(x,y) \middle| \begin{array}{l} y \in V : E(y,z) \\ \wedge d(x,y) \neq \perp \end{array} \right\} \\ \qquad\qquad\qquad\qquad\qquad\qquad \wedge x \neq z \wedge S \neq \emptyset. \end{array} \right\rfloor$$

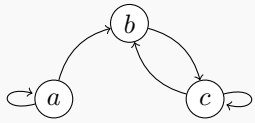
Note that whereas all examples of recursive definitions in Chapter 3 define total functions, in general, the defined distance-function is partial. In particular, if there is no path from a point x to a point y, then the distance between those points is undefined. As a result, the object space $\boldsymbol{\mathcal{O}_d}$ is now the set of all *partial* functions with signature $V \times V \to \mathbb{N}_\perp$. Additionally, the construction space $\boldsymbol{\mathcal{C}_d}$ is $\boldsymbol{\mathcal{O}_d}$ equipped with the definedness-order, i.e., $\boldsymbol{\mathcal{C}_d} = \langle (\mathbb{N}_\perp)^{V^2}, \leq_d \rangle$ where $\leq_d$ is the point-wise extension of the definedness order $\leq$ on $\mathbb{N}_\perp$. Again, as an indirect consequence of Proposition 6.3.5, $\boldsymbol{\mathcal{C}_d}$ is a bounded-complete cpo.

Analogously to the examples of recursive definitions in Chapter 3, the semantic operator corresponds to rule application. In other words, the semantic operator $O_d \colon \boldsymbol{\mathcal{C}_d} \to \boldsymbol{\mathcal{C}_d}$ is defined to map a function $f \in \boldsymbol{\mathcal{C}_d}$ to

$$O_d(f) := \begin{cases} (x,x) \mapsto & 0 \quad (\text{for } x \in V) \\ (x,z) \mapsto & \mathit{Min}_{\mathbb{N}}(S) + 1 \\ & \left(\begin{array}{l} \text{for } x,z \in V \text{ s.t. } x \neq z \text{ and } S \neq \emptyset \\ \text{and } S = \left\{ d(x,y) \middle| \begin{array}{l} y \in V : E(y,z) \\ \wedge d(x,y) \neq \perp \end{array} \right\} \end{array} \right) \end{cases}$$

Interestingly, this definition is defined over a well-founded order, i.e., if we are cautious not to apply any rules before we are sure they will remain valid, then rule application will find the intended fixpoint.

Example 6.3.13. *Consider the following simple graph:*



Now we can calculate the Well-founded fixpoint using the following well-founded induction of the ultimate approximator of O_d in fAFT, i.e., U_f: We start with an

application refinement

$$U_f(\boldsymbol{\mathcal{C}_d}) = \left\{ f \in \boldsymbol{\mathcal{C}_d} \middle| \begin{array}{ll} f(a,a) \in & \{0\}, \\ f(a,b) \in & \mathbb{N}_\perp \backslash \{0\}, \\ f(a,c) \in & \mathbb{N}_\perp \backslash \{0\}, \\ f(b,a) \in & \mathbb{N}_\perp \backslash \{0\}, \\ f(b,b) \in & \{0\}, \\ f(b,c) \in & \mathbb{N}_\perp \backslash \{0\}, \\ f(c,a) \in & \mathbb{N}_\perp \backslash \{0\}, \\ f(c,b) \in & \mathbb{N}_\perp \backslash \{0\}, \\ f(c,c) \in & \{0\} \end{array} \right\} = \boldsymbol{X}_1.$$

$\downarrow$ *application refinement*

$$U_f(\boldsymbol{X}_1) = \left\{ f \in \boldsymbol{\mathcal{C}_d} \middle| \begin{array}{ll} f(a,a) \in & \{0\}, \\ f(a,b) \in & \{1\}, \\ f(a,c) \in & \mathbb{N}_\perp \backslash \{0,1\}, \\ f(b,a) \in & \mathbb{N}_\perp \backslash \{0,1\}, \\ f(b,b) \in & \{0\}, \\ f(b,c) \in & \{1\}, \\ f(c,a) \in & \mathbb{N}_\perp \backslash \{0,1\}, \\ f(c,b) \in & \{1\}, \\ f(c,c) \in & \{0\} \end{array} \right\} = \boldsymbol{X}_3.$$

$\downarrow$ *application refinement*

$$U_f(\boldsymbol{X}_2) = \left\{ f \in \boldsymbol{\mathcal{C}_d} \middle| \begin{array}{ll} f(a,a) \in & \{0\}, \\ f(a,b) \in & \{1\}, \\ f(a,c) \in & \{2\}, \\ f(b,a) \in & \mathbb{N}_\perp \backslash \{0,1,2\}, \\ f(b,b) \in & \{0\}, \\ f(b,c) \in & \{1\}, \\ f(c,a) \in & \mathbb{N}_\perp \backslash \{0,1,2\}, \\ f(c,b) \in & \{1\}, \\ f(c,c) \in & \{0\} \end{array} \right\} = \boldsymbol{X}_3.$$

$$\downarrow \quad \textit{grounding refinement}$$

$$[\![\boldsymbol{X}_3^{\boldsymbol{\mathcal{L}}}, St^2_{U_f}(\boldsymbol{X}_3)]\!] = \left\{ \left\{ \begin{array}{ll} (a,a) & \mapsto \{0\}, \\ (a,b) & \mapsto \{1\}, \\ (a,c) & \mapsto \{2\}, \\ (b,a) & \mapsto \{\bot\}, \\ (b,b) & \mapsto \{0\}, \\ (b,c) & \mapsto \{1\}, \\ (c,a) & \mapsto \{\bot\}, \\ (c,b) & \mapsto \{1\}, \\ (c,c) & \mapsto \{0\} \end{array} \right\} \right. = \boldsymbol{X}_4.$$

$\boldsymbol{X}_4$ is an exact fixpoint of U_f. Therefore, $\boldsymbol{X}_4$ is the well-founded model. Thus, as promised, fAFT finds the correct model for this example. Notice that it is also the Kripke-Kleene model, as in the limit, applying the ultimate operator would have resulted in, e.g., $f(b,a) \in \mathbb{N}_\bot \setminus \mathbb{N} = \{\bot\}$.

In general, iteratively applying the ultimate approximator U_f yields a construction process with the following behavior: after i iterations, the function is uniquely determined for all pairs of nodes (x, y) whose shortest path has length less than or equal to $i - 1$. At the same time, for all other pairs with longer (or non-existent) paths, all values below $i - 1$ are ruled out as possible outcomes. Notice the crucial role of fine-grained upper bounds. Without the ability to rule out shorter paths, we could not always determine when a rule becomes permanently valid. Concretely, in the example above, within $\boldsymbol{X}_3$, the derivation that $d(a, c) = 2$ relies on the fact that we had excluded $d(a, c) = 0$; otherwise, $d(a, c)$ might just as well have been 1. This illustrates why relaxed iAFT would fail to recover the intended model in this example: it cannot discriminate between different sets of possible values. Instead, a function in relaxed iAFT is either fully defined on a domain element or left entirely undefined. In other words, although relaxed iAFT is applicable on bounded-complete cpos, the inherently more natural fit of fAFT carries undeniable benefits.

However, shifting to non-monotonic operators without an underlying well-founded order, reveals the limits of fAFT: despite it surpassing (relaxed) iAFT in expressiveness and usability, it still does not always provide sufficient expressive power. Although fAFT seems perfectly suited for the construction space of recursive definitions as proposed in Chapter 3, oftentimes for more intricate recursive definition the well-

founded fixpoint will not be exact. This is due to the fact that the lower bound cannot distinguish between a value being fully undefined and a value for which we know that its defined (and we might know some restrictions on the possible values), but we do not know the exact value. This suggests that, in general, non-monotonic recursive definitions require an even more precise approximation space. To illustrate this, consider the following variant of the previous definition where edges are assigned weights:

Example 6.3.14. *Imagine a game where you are driving a vehicle. Depending on which roads you take, your vehicle gets damaged or is repaired. Let us represent the roads by a weighted directed graph $\mathcal{G} = (V, E, w)$ with vertices V, edges E and a weight function w that maps every edge to a integer. If $w(x, y)$ is positive, then $w(x, y)$ represents the damage you suffered from taking the road (x, y). If $w(x, y)$ is negative, then $w(x, y)$ represents how much your damage can be repaired by taking the road (x, y). The minimal damage between point $x \in V$ and point $y \in V$ is defined as the least possible damage the vehicle can have when starting in x and ending in y. We assume that the damage can never go below 0. Initially, the car starts with an arbitrary amount of damage D. One can formally define this damage function $d' : V \times V \to \mathbb{N}_\perp$ as follows:*

$$\left\lfloor \begin{array}{ll} \forall x \in V : d'(x,x) := Min_{\mathbb{N}}(\{D\} \cup S) & \leftarrow S = \{Max_{\mathbb{N}}(\{0, d'(x,y) + w(y,x)\}) \mid \\ & \qquad y \in V : d'(x,y) \neq \perp \wedge E(y,x)\}. \\ \forall x, z \in V : d'(x,z) := Min_{\mathbb{N}}(S) & \leftarrow S = \{Max_{\mathbb{N}}(\{0, d'(x,y) + w(y,z)\}) \mid \\ & \qquad y \in V : d'(x,y) \neq \perp \wedge E(y,z)\} \\ & \qquad \wedge S \neq \emptyset \wedge x \neq z. \end{array} \right\rfloor$$

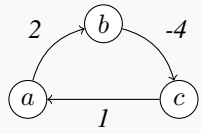
In each rule, the set S represents the resulting damages of extensions of the best paths between x and y for which the minimal damage is well-defined. The first rule considers the case where $x = z$ and assigns the minimal value of such extension provided that it is better than the initial damage. In other words, the first rule ensures that the minimal damage to go from x to x is at most the initial damage, which can be obtained by taking no edges. On the other hand, the second rule identifies the minimal damage for any other pair of vertices (x, z) provided that there is at least a path from x to z for which the damage is well-defined, i.e., $S \neq \emptyset$.

As the examples are strongly related, the object space and the construction space correspond to those of Example 5.1.1. The semantic operator $O_{d'} : \boldsymbol{\mathcal{C}_{d'}} \to \boldsymbol{\mathcal{C}_{d'}}$ is

defined to map a function $f \in \mathcal{C}_{d'}$ to

$$O_{d'}(f) := \begin{cases} (x,x) \mapsto & Min_{\mathbb{N}}(\{D\} \cup S) \\ & \left(\begin{array}{l} \text{for } x \in V \\ \text{and } S = \{Max_{\mathbb{N}}(\{0, d'(x,y) + w(y,x)\}) \mid \\ \qquad\qquad y \in V : d'(x,y) \neq \bot \wedge E(y,x)\} \end{array} \right) \\ (x,z) \mapsto & Min_{\mathbb{N}}(S) \\ & \left(\begin{array}{l} \text{for } x, z \in V \text{ s.t. } x \neq z \text{ and } S \neq \emptyset \\ \text{and } S = \{Max_{\mathbb{N}}(\{0, d'(x,y) + w(y,z)\}) \mid \\ \qquad\qquad y \in V : d'(x,y) \neq \bot \wedge E(y,z)\} \end{array} \right) \end{cases}$$

Example 6.3.15. *Consider the following simple graph where the labels on the edges represent the weight function w:*



Moreover, let us assume that $D = 4$.

Now we can calculate the Kripke-Kleene and Well-founded fixpoints using the ultimate approximator of $O_{d'}$ in fAFT, i.e., U_f^*. For any $f \in \mathcal{C}_{d'}$, we can make the following observations about $O_{d'}(f)$:

- *The minimal damage to go from a point $x \in V$ to itself is always well-defined, i.e., $(O_{d'}(f))(x,x) \neq \bot$. Moreover, it can never be more than the initial damage, i.e., $(O_{d'}(f))(x,x) \leq D = 4$. In summary, for every $x \in V$, $(O_{d'}(f))(x,x) \in \{0,1,2,3,4\}$.*
- *If the minimal damage to go from any point $x \in V$ to a is well-defined, then it must be at least 1, since the only incoming edge to node a has weight 1, i.e., for every $x \in V$, $(O_{d'}(f))(x,a) \in \mathbb{N}_\bot \backslash \{0\}$.*
- *If the minimal damage to go from any point $x \in V$ to b is well-defined, then it must be at least 2, since the only incoming edge to node a has weight 2, i.e., for every $x \in V$, $(O_{d'}(f))(x,b) \in \mathbb{N}_\bot \backslash \{0,1\}$.*

As a result, we see that

$$U_f^*(\mathcal{C}_{d'}) = \left\{ f \in \mathcal{C}_{d'} \middle| \begin{array}{ll} f(a,a) \in & \{\bot,1,2,3,4\}, \\ f(a,b) \in & \mathbb{N}_\bot \backslash \{0,1\}, \\ f(a,c) \in & \mathbb{N}_\bot, \\ f(b,a) \in & \mathbb{N}_\bot \backslash \{0\}, \\ f(b,b) \in & \{\bot,2,3,4\}, \\ f(b,c) \in & \mathbb{N}_\bot, \\ f(c,a) \in & \mathbb{N}_\bot \backslash \{0\}, \\ f(c,b) \in & \mathbb{N}_\bot \backslash \{0,1\}, \\ f(c,c) \in & \{\bot,0,1,2,3,4\} \end{array} \right\} = \boldsymbol{X}_1.$$

Note that $\bot$ reappears in the possible images of (a,a), (b,b) and (c,c) due to the fact that flowers must include their greatest lower bound. Now, for any $f \in \boldsymbol{X}_1$, we can make additional observations about $O_{d'}(f)$. For example, for the minimal damage to go from a to b to be well-defined, it must be obtained by $f(a,a)+2$ where $f(a,a)$ is well-defined. By assumption, $f(a,a) \in \{\bot,1,2,3,4\}$, thus $(O_{d'}(f))(a,b) \in \{\bot,3,4,5,6\}$. In each step of the fastest monotonic induction process we are able to further refine our approximant using similar observations.

$$U_f^*(\boldsymbol{X}_1) = \left\{ f \in \mathcal{C}_{d'} \middle| \begin{array}{ll} f(a,a) \in & \{\bot,1,2,3,4\}, \\ f(a,b) \in & \{\bot,3,4,5,6\}, \\ f(a,c) \in & \mathbb{N}_\bot, \\ f(b,a) \in & \mathbb{N}_\bot \backslash \{0\}, \\ f(b,b) \in & \{\bot,2,3,4\}, \\ f(b,c) \in & \{\bot,0\}, \\ f(c,a) \in & \{\bot,1,2,3,4,5\}, \\ f(c,b) \in & \mathbb{N}_\bot \backslash \{0,1\}, \\ f(c,c) \in & \{\bot,0,1,2,3,4\} \end{array} \right\} = \boldsymbol{X}_2.$$

$$\downarrow$$

$$U_f^*(\boldsymbol{X}_2) = \left\{ f \in \mathcal{C}_{d'} \middle| \begin{array}{ll} f(a,a) \in & \{\bot,1,2,3,4\}, \\ f(a,b) \in & \{\bot,3,4,5,6\}, \\ f(a,c) \in & \{\bot,0,1,2\}, \\ f(b,a) \in & \{\bot,1\}, \\ f(b,b) \in & \{\bot,2,3,4\}, \\ f(b,c) \in & \{\bot,0\}, \\ f(c,a) \in & \{\bot,1,2,3,4,5\}, \\ f(c,b) \in & \{\bot,3,4,5,6,7\}, \\ f(c,c) \in & \{\bot,0,1,2,3,4\} \end{array} \right\} = \boldsymbol{X}_3.$$

$$\downarrow$$

$$U_f^*(\boldsymbol{X}_3) = \left\{ f \in \mathcal{C}_{d'} \middle| \begin{array}{ll} f(a,a) \in & \{\bot,1,2,3,4\}, \\ f(a,b) \in & \{\bot,3,4,5,6\}, \\ f(a,c) \in & \{\bot,0,1,2\}, \\ f(b,a) \in & \{\bot,1\}, \\ f(b,b) \in & \{\bot,3,4\}, \\ f(b,c) \in & \{\bot,0\}, \\ f(c,a) \in & \{\bot,1,2,3,4,5\}, \\ f(c,b) \in & \{\bot,3,4,5,6,7\}, \\ f(c,c) \in & \{\bot,0,1,2,3,4\} \end{array} \right\} = \boldsymbol{X}_4.$$

$\boldsymbol{X}_4$ is a fixpoint of U_f^*. Therefore $KK(U_f^*) = \boldsymbol{X}_4$. Clearly this is not an exact fixpoint. Furthermore, we notice that $\boldsymbol{X}_4^{\mathcal{L}} = f_\bot$, i.e., the function that maps every pair of vertices to $\bot$. As a result, $St^1(\mathcal{C}_{d'}) = f_\bot$. On the other hand we find

$$U_f^*(\{f_\bot\}) = \left\{ f \in \mathcal{C}_{d'} \middle| \begin{array}{ll} f(x,x) = 4 & \textit{(if } x \in V\textit{)} \\ f(x,y) = \bot & \textit{(if } x \neq y\textit{)} \end{array} \right\} = \boldsymbol{Y}_1$$

$$\downarrow$$

$$U_f^*({\boldsymbol{Y}_1}^{\downarrow}) = \left\{ f \in \mathcal{C}_{d'} \middle| \begin{array}{ll} f(x,x) = 4 & \textit{(if } x \in V\textit{)} \\ f(a,b) \in \{\bot,6\} & \\ f(b,c) \in \{\bot,0\} & \\ f(c,a) \in \{\bot,5\} & \end{array} \right\} = \boldsymbol{Y}_2$$

$$\downarrow$$

$$U_f^*(\boldsymbol{Y}_2{}^{\downarrow}) = \left\{ f \in \boldsymbol{\mathcal{C}_{d'}} \;\middle|\; \begin{array}{ll} f(x,x) = 4 & \textit{(if } x \in V\textit{)} \\ f(a,b) \in \{\bot, 6\} & \\ f(a,c) \in \{\bot, 2\} & \\ f(b,a) \in \{\bot, 1\} & \\ f(b,c) \in \{\bot, 0\} & \\ f(c,a) \in \{\bot, 5\} & \\ f(c,b) \in \{\bot, 7\} & \end{array} \right\} = \boldsymbol{Y}_3$$

$$\downarrow$$

$$U_f^*(\boldsymbol{Y}_3{}^{\downarrow}) = \left\{ f \in \boldsymbol{\mathcal{C}_{d'}} \;\middle|\; \begin{array}{ll} f(x,x) \in \{\bot, 4, 3\} & \textit{(if } x \in V\textit{)} \\ f(a,b) \in \{\bot, 6\} & \\ f(a,c) \in \{\bot, 2\} & \\ f(b,a) \in \{\bot, 1\} & \\ f(b,c) \in \{\bot, 0\} & \\ f(c,a) \in \{\bot, 5\} & \\ f(c,b) \in \{\bot, 7\} & \end{array} \right\} = \boldsymbol{Y}_4$$

$$\downarrow$$

$$\vdots$$

$$\downarrow$$

$$U_f^*(\boldsymbol{Y}_{15}{}^{\downarrow}) = \left\{ f \in \boldsymbol{\mathcal{C}_{d'}} \;\middle|\; \begin{array}{l} f(a,a) \in \{\bot, 4, 3, 2, 1\} \\ f(a,b) \in \{\bot, 6, 5, 4, 3\} \\ f(a,c) \in \{\bot, 2, 1, 0\} \\ f(b,a) \in \{\bot, 1\} \\ f(b,b) \in \{\bot, 4, 3\} \\ f(b,c) \in \{\bot, 0\} \\ f(c,a) \in \{\bot, 5, 4, 3, 2, 1\} \\ f(c,b) \in \{\bot, 7, 6, 5, 4, 3\} \\ f(c,c) \in \{\bot, 4, 3, 2, 1, 0\} \end{array} \right\} = \boldsymbol{X}_4$$

In other words, $St(\boldsymbol{\mathcal{C}_{d'}}) = \boldsymbol{X}_4$. *As* $\boldsymbol{X}_4$ *is a fixpoint of* U_f^*, *it is clear that* $St^1(\boldsymbol{X}_4) = f_\bot$. *Moreover, as* $\boldsymbol{X}_4^{\boldsymbol{\mathcal{L}}} = f_\bot$, *this entails that* $St^2(\boldsymbol{X}_4) = St^2(\boldsymbol{\mathcal{C}_{d'}}) = \boldsymbol{X}_4^{\boldsymbol{\mathcal{U}}}$. *In summary,*

$$WF(U_f^*) = \boldsymbol{X}_4 = KK(U_f^*),$$

thus clearly, the approximation space $\boldsymbol{\mathcal{A}_f}(\boldsymbol{\mathcal{C}_{d'}})$ *is not refined enough to find the intended fixpoint with the well-founded semantics.*

Interestingly, as in Example 5.1.3, this example features an operator for which the iteration of $O_{d'}$ converges to the intended fixpoint regardless of the starting exact element (function). Therefore, it is difficult to argue that this operator is so poorly behaved that our framework could not be expected to find the correct solution.

Specifically, the problem lies with the lack of refined lower bounds which prevents fAFT from finding the intended fixpoint for this non-monotonic recursive definition. Hence, solving such definitions requires an even more expressive and more precise approximation space. Specifically, we require more precise lower bounds. As a result, the needed approximation space is not non-degenerate. For example, we could obtain more precise lower bounds by using set-valued functions that keep track of the possible values. In such space, a function that maps a domain element x to a set $S \subseteq B$ then denotes the knowledge that we know that the defined function is well-defined for x and must map to something in S without specifying the exact value it maps to. The bigger the set, the less defined the function is for x. Despite its non-degeneracy, most natural approximators on this space successfully capture the intended semantics.

In summary, we have identified two fairly natural approximation spaces for bounded-complete cpos. The first, and most intuitively fitting, is that of fAFT, while the second is a relaxed version of iAFT. Both spaces are applicable, but fAFT clearly offers advantages over the relaxed iAFT, both for monotonic and non-monotonic operators. In particular, the imprecise upper bounds in the relaxed iAFT prevent it from providing the usual benefits associated with AFT.

At the same time, this section highlights a gap in the framework. Even fAFT may fail on certain well-behaved non-monotonic recursive definitions, including cases with a unique fixpoint. The issue lies with the imprecise lower bounds. However, refining them to capture the intended semantics would compromise non-degeneracy, leaving an open problem for future investigation.

6.4 fAFT and Normal Logic Programming

Having explored fAFT for constructive definitions, we now turn our attention back to logic programming. In Chapter 4, we considered logic programs within the interval-based setting of iAFT, where approximants were represented as pairs of two-valued interpretations. In this section, we revisit the same domain but apply the flower-based approach instead. For brevity and simplicity, we focus here on normal logic programming (LP), but the ideas can be extended to extensions of LP.

As explained in Chapter 4, the exact space typically used for LP is given by the set of interpretations over a vocabulary of ground atoms Σ ordered by the point-wise truth order. Oftentimes a two-valued interpretation $\mathcal{I}$ over Σ is represented by the set of atoms that are true in $\mathcal{I}$, entailing an exact space $\langle 2^\Sigma, \subseteq \rangle$. Remember from Chapter 4 that for a logic program P consisting of rules r of the form $(p \leftarrow \psi)$ such that p is a propositional atom and ψ is a formula, the semantic operator is given by the immediate consequence operator $T_P : 2^\Sigma \to 2^\Sigma$ which maps any interpretation $\mathcal{I}$ to an

interpretation $\mathcal{J}$ such that for every ground atom p,

$$\mathcal{H}^2_{\mathcal{J}}(p) = \text{lub}_{\leq}(\{\mathcal{H}^2_{\mathcal{I}}(\psi) \mid (p \leftarrow \psi) \in P\})$$

where $\mathcal{H}^2_I$ is the two-valued truth function for $\mathcal{I}$.

Chapter 4 identified two different types of approximators for the immediate consequence operator. The first was the ultimate approximator, i.e., the most precise approximator. We know that fAFT provides an ultimate approximator as well, by Proposition 5.8.6. While the ultimate approximator is interesting because it explores the limit of the expressiveness of an approximation space for a given semantic operator, it is generally not the most natural approximator for a logic program. The ASP and LP community seemingly prefer semantics associated with the other type of approximators, i.e., extensions of Fitting's three-valued immediate consequence operator Φ^{SK}_P based on a regular three-valued truth-function. Specifically, the ultimate approximator is defined semantically rather than syntactically, i.e., the ultimate approximator is defined in terms of the behavior of the semantic operator T_P. In comparison, Φ^{SK}_P is defined syntactically, i.e., the approximator can be derived straightforwardly from the syntax of the rules in the program. Notably, a syntactical approximator is generally more strict than the ultimate operator, i.e., it ensures that if we at some point in our construction process derive the truth of a ground atom p using a rule $(p \leftarrow \psi)$, then this rule will remain applicable in any subsequent step. In other words, it not only guarantees consistency of the model itself, but also of the explanation for the model. Moreover, this operator was computationally tractable, striking a balance between interpretability, correctness, and efficiency.

In contrast, when moving to fAFT, it is unlikely that a similarly tractable operator can be achieved. The reason is inherent to the richer structure of the approximation space: upper bounds are now represented by sets of elements rather than single elements, which we expect to be reflected in the computational complexity of any corresponding operator, at least in the general case, and especially when the additional expressive power of the space is exploited.

Therefore, our focus shifts primarily to the explainability aspect of the operator. Our goal is to identify an approximator Γ_P that maintains the intuitive, compositional nature of Φ^{SK}_P while taking advantage of the improved accuracy provided by fAFT. Such operator should remain as syntactic and rule-driven as possible, facilitating a direct correspondence between program structure and construction. In particular, we aim for an operator that supports constancy of the rules we use as explanations for the truth of ground atoms throughout the construction, ensuring that the underlying reasoning can be easily interpreted.

However, when fully exploiting the additional precision provided by fAFT, our proposed approximator Γ_P may become difficult to interpret after all. To address this, we subsequently introduce in Section 6.4.2 a series of condensed variants that

seek to balance interpretability and efficiency on the one hand with accuracy and expressiveness on the other.

6.4.1 Ultimate Conjunctional Immediate Consequence Operator

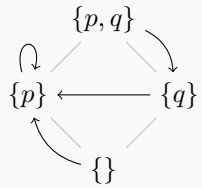
To develop such an operator, we look at compositionality, syntactic transparency, and preservation of explanatory consistency as the foundation for our present extension of Φ_P^{SK} to the setting of fAFT. The key challenge lies in adapting these ideas to an approximation space where upper bounds are represented by sets of elements rather than single elements, while still retaining a clear correspondence between syntactic rules and semantic construction steps.

Let us begin by introducing the example that will serve as our running illustration throughout this section. Consider the following simple logic program where the operator has only one fixpoint. The well-founded semantics of both iAFT-ultimate approximator and Φ_P^{SK} correctly identify this fixpoint as the well-founded model. However, if we use flowers, the fAFT-ultimate approximator already finds the intended fixpoint using the Kripke-Kleene semantics.

Example 6.4.1. *Consider the following simple program P over $\Sigma = \{p, q\}$:*

$$q \leftarrow p \wedge q.$$
$$p \leftarrow \neg q.$$
$$p \leftarrow \neg p.$$

This program entails the following simple operator T_P on 2^Σ:



Before constructing the operator Γ_P, we first need to understand the nature of the precision gain that fAFT offers compared to iAFT. To this end, we take a closer look at the implications of the *ALBs* and *AUBs* in fAFT approximation spaces, and how they relate to the lower and upper bounds in iAFT. In particular the *AUBs* are of interest. The *AUB* $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$ of an approximant represents that at most $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$ is in the approximant, anything greater is not considered. In the case of flowers, an *AUB* is an anti-chain of elements

in the exact space. Relating back to the space of intervals, we can look at such upper bound in two ways. On the one hand, we can treat a flower as a union of intervals. In this sense we think of the newly added flowers as intervals to which we added elements. For example, if a flower $\boldsymbol{X}$ of interpretations has the *AUB* $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}} = \{\{p\}, \{q\}\}$, then we consider all interpretations where at most p is true, in addition to those where at most q is true. However, the *AUB* is typically used to exclude elements from the approximant in each iteration step. Therefore it makes sense to think of flowers as intervals from which we removed a set of elements. For example, according to this view, the *AUB* $\boldsymbol{X}^{\boldsymbol{\mathcal{U}}} = \{\{p\}, \{q\}\}$ then corresponds to the set of interpretations where at most p and q are true from which we deleted the interpretations where both p and q are true.

The latter point of view suggests that to exploit the richness of fAFT we need explicit information about conjunctions of propositional atoms instead of deriving their truth based on the truth-value of their conjuncts. Since our objective is for Γ_P to remain as rule-driven and syntactically transparent as possible, we propose to treat conjunctions as first-class entities within the program. Concretely, we do so by (1) adding new propositional atoms which represent the different possible conjunctions over Σ, (2) adding new rules in the program for each of these newly added atoms by combining different rules for the different conjuncts, and (3) replacing every occurrence of a conjunction by the corresponding newly added atom. Interestingly, this transformation leaves the rules for the original ground atoms virtually unchanged, it primarily introduces additional rules that explicitly define the truth of conjunctions. Consequently, the way in which we derive and explain the truth of ground atoms remains unaffected. In this sense, our explanations for ground atoms are preserved.

The key difference lies in how conjunctions are handled. Instead of determining their truth values compositionally from those of their conjuncts, we now derive them by identifying compatible combinations of rules that support the truth of the conjuncts. This shift allows us to exploit the additional expressive power provided by the fAFT space. However, this also means that some of the immediate syntactic transparency of conjunctions may be lost, as their truth is now derived indirectly via these combinations.

Nonetheless, this trade-off affects only the interpretation of conjunctions, not of individual ground atoms. The operator retains its full interpretability and explanatory power with respect to ground atoms, ensuring that the central goal of interpretability and explainability remains intact.

Definition 6.4.2. *Let Σ be a finite set of propositional atoms, the* conjunctional extension of Σ*, denoted by $\Sigma^{\wedge}$ is a set of propositional atoms*

$$\Sigma^{\wedge} = \Sigma \cup \{c_S \mid S \subseteq \Sigma \wedge |S| > 1\}$$

where c_S are new propositional atoms.

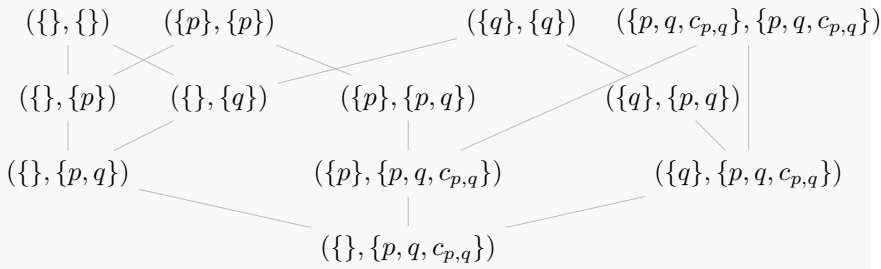
To simplify the discussion and the following formal definitions, we will sometimes use the analogous notation $c_{\{p\}}$ to denote a propositional atom $p \in \Sigma$ when discussed in the context of $\Sigma^\wedge$. Informally, a propositional atom c_S represents the conjunction of all conjuncts in S. For example $c_{\{K_q, K_r\}}$ represents the conjunction $K_q \wedge K_r$, i.e., if $c_{\{K_q,K_r\}}$ evaluates to true in a two-valued interpretation $\mathcal{I}^\wedge$ over $\Sigma^\wedge$, then this can be obtained from a two-valued interpretation $\mathcal{I}$ over Σ in which $K_q \wedge K_r$ is true, and analogously for the case where $c_{\{K_q,K_r\}}$ evaluates to false. Note that, in the context of aforementioned relation between interpretations over Σ and $\Sigma^\wedge$, it does not make sense to have an interpretation $\mathcal{I}^\wedge$ over $\Sigma^\wedge$ such that for some $S \subseteq \Sigma$, c_S is true in $\mathcal{I}^\wedge$ while for a subset $S' \subseteq S$, $c_{S'}$ is not true. Analogously, we can define a subset of sensible three-valued interpretations over $\Sigma^\wedge$. Remember from Chapter 4 that a three-valued interpretation $\mathfrak{A}^\wedge$ can be represented by a pair of two-valued interpretations $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ where $\mathcal{I}^\wedge = \{c_S \in \Sigma^\wedge \mid \mathcal{H}^{SK}_{\mathfrak{A}^\wedge}(c_S) = \mathbf{t}\}$ and $\mathcal{J}^\wedge = \{c_S \in \Sigma^\wedge \mid \mathcal{H}^{SK}_{\mathfrak{A}^\wedge}(c_S) \in \{\mathbf{t}, \mathbf{u}\}\}$.

Definition 6.4.3. *Let Σ be a finite set of propositional atoms and let $\Sigma^\wedge$ be the conjunctional extension of Σ. A three-valued interpretation $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is* coherent *with respect to Σ if for every $c_S \in \Sigma^\wedge$,*

- *$c_S \in \mathcal{I}^\wedge$ iff $S \subseteq \mathcal{I}^\wedge$, and*
- *if $c_S \in \mathcal{J}^\wedge$, then for every $\emptyset \subsetneq S' \subseteq S$, $c_{S'} \in \mathcal{J}^\wedge$.*

Example 6.4.4 (Example 6.4.1, contd.)**.** *Let $\Sigma = \{p, q\}$, then $\Sigma^\wedge = \{p, q, c_{\{p,q\}}\}$.*

In this context, the set of three-valued interpretations that are coherent with respect to Σ gives rise to the following poset when ordered by the precision order:



An attentive reader will notice the strong resemblance between the poset above and the decomposition spaces $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$. In fact, for every finite set Σ, the poset of three-valued interpretations over $\Sigma^\wedge$ ordered by the precision-order is isomorphic to the consistent part of the product-space $\boldsymbol{\mathcal{L}_f} \times \boldsymbol{\mathcal{U}_f}$. In other words, for normal

logic programs, three-valued interpretations over the conjunctional extension Σ give rise to an alternative pair of natural decomposition spaces for flowers.

With this in mind, just as for regular three-valued interpretations over Σ and intervals, we notice a compelling connection between coherent three-valued interpretations $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ over $\Sigma^\wedge$ and flowers of two-valued interpretations over Σ.

Definition 6.4.5. *Let P be a normal logic program over a finite set of propositional atoms Σ. Let $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$ be the lower and upper decomposition spaces for $\mathcal{F}_{2^\Sigma}$ as defined in Example 5.4.2, we define the functions $\psi_{\boldsymbol{\mathcal{L}}} : \boldsymbol{\mathcal{L}_f} \to \{\mathcal{I}^\wedge \mid (\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent$\}$ and $\psi_{\boldsymbol{\mathcal{U}}} : \boldsymbol{\mathcal{U}_f} \to \{\mathcal{J}^\wedge \mid (\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent$\}$ such that for every $l \in \boldsymbol{\mathcal{L}_f}$ and $u \in \boldsymbol{\mathcal{U}_f}$:*

$$\psi_{\boldsymbol{\mathcal{L}}}(l) = \{c_S \mid S \subseteq l\}$$

$$\psi_{\boldsymbol{\mathcal{U}}}(u) = \{c_S \mid \exists S' \in u : S \subseteq S'\}$$

Note that we can rewrite both $\psi_{\boldsymbol{\mathcal{L}}}$ and $\psi_{\boldsymbol{\mathcal{U}}}$ as an operator that maps a $b \in \boldsymbol{\mathcal{L}_f}$ or $b \in \boldsymbol{\mathcal{U}_f}$ to $\{c_S \mid S \in b^\downarrow\}$.

Proposition 6.4.6. *$\psi_{\boldsymbol{\mathcal{L}}}$ is an isomorphism.*

Proof. Let $l_1, l_2 \in \boldsymbol{\mathcal{L}_f}$. It is trivial to see that if $l_1 \neq l_2$, then $l_1 \backslash l_2 \neq \emptyset$ or $l_2 \backslash l_1 \neq \emptyset$. Let us assume $s \in l_1 \backslash l_2$, then $s = c_{\{s\}} \in \psi_{\boldsymbol{\mathcal{L}}}(l_1) \backslash \psi_{\boldsymbol{\mathcal{L}}}(l_2)$, and similarly for the other case. Therefore, if $l_1 \neq l_2$, then $\psi_{\boldsymbol{\mathcal{L}}}(l_1) \neq \psi_{\boldsymbol{\mathcal{L}}}(l_2)$. Hence, $\psi_{\boldsymbol{\mathcal{L}}}$ is injective.

Let $\mathcal{I}^\wedge \in \{\mathcal{I}^\wedge \mid (\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent$\}$. It is easy to see that $\mathcal{I}^\wedge \cap \Sigma \in \boldsymbol{\mathcal{L}_f}$. We will now show that $\mathcal{I}^\wedge = \psi_{\boldsymbol{\mathcal{L}}}(\mathcal{I}^\wedge \cap \Sigma)$. Assume $c_S \in \mathcal{I}^\wedge$. Since $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent, $c_S \in \mathcal{I}^\wedge$ iff $S \subseteq \mathcal{I}^\wedge$. Moreover, $S \subseteq \Sigma$ by definition of $\Sigma^\wedge$. Hence, $c_S \in \mathcal{I}^\wedge$ iff $S \subseteq \mathcal{I}^\wedge \cap \Sigma$. By definition of $\psi_{\boldsymbol{\mathcal{L}}}$, this is equivalent to $c_S \in \psi_{\boldsymbol{\mathcal{L}}}(\mathcal{I}^\wedge \cap \Sigma)$. As a result, every $\mathcal{I}^\wedge \in \{\mathcal{I}^\wedge \mid (\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent$\}$ is the image of at least one element $l \in \boldsymbol{\mathcal{L}_f}$, i.e., $\psi_{\boldsymbol{\mathcal{L}}}$ is surjective and thus bijective.

Finally, using the rewrite $\psi_{\boldsymbol{\mathcal{L}}}(l) = \{c_S \mid S \in l^\downarrow\}$, it trivially follows that if $l_1 \preceq_f l_2$, i.e., ${l_1}^\downarrow \subseteq {l_2}^\downarrow$, then $\psi_{\boldsymbol{\mathcal{L}}}(l_1) \subseteq \psi_{\boldsymbol{\mathcal{L}}}(l_2)$. Hence $\psi_{\boldsymbol{\mathcal{L}}}$ is order-preserving. □

Proposition 6.4.7. *$\psi_{\boldsymbol{\mathcal{U}}}$ is an isomorphism.*

Proof. Let $u_1 \neq u_2$. Trivially, $u_1 \backslash u_2 \neq \emptyset$ or $u_2 \backslash u_1 \neq \emptyset$. Let us assume there exists an interpretation $\mathcal{I} \in u_1 \backslash u_2$. Clearly, by definition of $\psi_{\boldsymbol{\mathcal{U}}}$, $c_{\mathcal{I}} \in \psi_{\boldsymbol{\mathcal{U}}}(u_1)$. If $c_{\mathcal{I}} \notin \psi_{\boldsymbol{\mathcal{U}}}(u_2)$, then $\psi_{\boldsymbol{\mathcal{U}}}(u_1) \neq \psi_{\boldsymbol{\mathcal{U}}}(u_2)$. Otherwise, there must exist an $S \in u_2$ such that $\mathcal{I} \subseteq S$. Since $\mathcal{I} \in u_1 \backslash u_2$, we know that $\mathcal{I} \neq S$. Additionally, $c_S \in \psi_{\boldsymbol{\mathcal{U}}}(u_2)$. However, since u_1 is an anti-chain, it is impossible that there exists an interpretation $\mathcal{I}' \in u_1$

such that $\mathcal{I} \subsetneq S \subseteq \mathcal{I}'$, hence $c_S \notin \psi_{\boldsymbol{\mathcal{U}}}(u_1)$. Thus, in any case, if $u_1 \neq u_2$, then $\psi_{\boldsymbol{\mathcal{U}}}(u_1) \neq \psi_{\boldsymbol{\mathcal{U}}}(u_2)$. In summary, $\psi_{\boldsymbol{\mathcal{U}}}$ is injective.

Let $\mathcal{J}^\wedge \in \{\mathcal{J}^\wedge \mid (\mathcal{I}^\wedge, \mathcal{J}^\wedge) \text{ is coherent}\}$. Let $u = \{S \mid c_S \in \mathcal{J}^\wedge \wedge \neg\exists S' \subseteq \Sigma : c_{S'} \in \mathcal{J}^\wedge \wedge S \subsetneq S'\}$. It is easily verified that $u \in \boldsymbol{\mathcal{U}_f}$, since by choice of u, u must be an anti-chain. Moreover, Σ is closed, hence $u^\downarrow$ is closed as well. We will now show that $\mathcal{J}^\wedge = \psi_{\boldsymbol{\mathcal{U}}}(u)$. Assume $c_S \in \mathcal{J}^\wedge$. Take a saturated chain $C \subseteq \{S \mid c_S \in \mathcal{J}^\wedge\}$ such that $S \in C$. Since $C \subseteq \Sigma^\wedge$ and $\Sigma^\wedge$ is finite, C must be finite as well. Hence, $\mathrm{lub}(C) \in C$, and thus $c_{\mathrm{lub}(C)} \in \mathcal{J}^\wedge$. Furthermore, $S \subseteq \mathrm{lub}(C)$, thus, by definition of $\psi_{\boldsymbol{\mathcal{U}}}$, $c_S \in \psi_{\boldsymbol{\mathcal{U}}}(u)$. On the other hand, assume $c_S \in \psi_{\boldsymbol{\mathcal{U}}}(u)$. By definition of $\psi_{\boldsymbol{\mathcal{U}}}$, there must exist an interpretation $\mathcal{I} \in u$ such that $S \subseteq \mathcal{I}$. Then by choice of u, $c_{\mathcal{I}} \in \mathcal{J}^\wedge$. Furthermore, since $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent and $S \subseteq \mathcal{I}$, this entails that $c_S \in \mathcal{J}^\wedge$. Thus, $\mathcal{J}^\wedge = \psi_{\boldsymbol{\mathcal{U}}}(u)$. As a result, we can conclude that $\psi_{\boldsymbol{\mathcal{U}}}$ is surjective, and therefore a bijection.

Finally, it is trivial to see that $\psi_{\boldsymbol{\mathcal{U}}}$ is order-preserving. If $u_1 \preceq_f u_2$, i.e., ${u_1}^\downarrow \subseteq {u_2}^\downarrow$, then $\psi_{\boldsymbol{\mathcal{U}}}(u_1) \subseteq \psi_{\boldsymbol{\mathcal{U}}}(u_2)$. □

As a result, we can conclude that $\{\mathcal{I}^\wedge \mid (\mathcal{I}^\wedge, \mathcal{J}^\wedge) \text{ is coherent}\}$ and $\{\mathcal{J}^\wedge \mid (\mathcal{I}^\wedge, \mathcal{J}^\wedge) \text{ is coherent}\}$ can be viewed as alternative representations of the decomposition spaces we know for flowers of interpretations over Σ. As a final missing piece of the puzzle, we define the isomorphic version of the recomposition function for flowers.

Definition 6.4.8. *Let Σ be a finite set of propositional atoms and let $\Sigma^\wedge$ be the conjunctional extension of Σ. We define t^f as the function which maps a coherent three-valued interpretation $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ to the set of all two-valued interpretations $\mathcal{I}$ over Σ satisfying*

- $(\mathcal{I}^\wedge \cap \Sigma) \subseteq \mathcal{I}$, *and*
- $c_{\mathcal{I}} \in \mathcal{J}^\wedge$.

Proposition 6.4.9. *Let P be a normal logic program over a finite set of propositional atoms Σ. Let $\boldsymbol{\mathcal{L}_f}$ and $\boldsymbol{\mathcal{U}_f}$ be the lower and upper decomposition spaces for $\mathcal{F}_{2^\Sigma}$ with recomposition function $[\![.,.]\!]$ as defined in Example 5.4.2. If $l \in \boldsymbol{\mathcal{L}_f}$ and $u \in \boldsymbol{\mathcal{U}_f}$, then $[\![l, u]\!] = t^f(\psi_{\boldsymbol{\mathcal{L}}}(l), \psi_{\boldsymbol{\mathcal{U}}}(u))$.*

Proof. Let $\psi_{\boldsymbol{\mathcal{L}}}(l) = \mathcal{I}^\wedge$ and let $\psi_{\boldsymbol{\mathcal{U}}}(u) = \mathcal{J}^\wedge$.

Let $\mathcal{I} \in [\![l, u]\!] = l^\uparrow \cap u^\downarrow$. Then $l \subseteq \mathcal{I}$. Moreover, for every $c_S \in \mathcal{I}^\wedge$, $S \subseteq l$. Thus, $S \subseteq \mathcal{I}$. Consequently, $\mathcal{I}^\wedge \cap \Sigma \subseteq \mathcal{I}$. Furthermore, $\mathcal{I} \in u^\downarrow$. Thus $c_{\mathcal{I}} \in \mathcal{J}^\wedge$, by definition of $\psi_{\boldsymbol{\mathcal{U}}}$. Consequently, clearly $\mathcal{I} \in t^f(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$.

Similarly, let $\mathcal{I} \in t^f(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$. Then $l = \psi_{\boldsymbol{\mathcal{L}}}^{-1}(\mathcal{I}^\wedge) = (\mathcal{I}^\wedge \cap \Sigma) \subseteq \mathcal{I}$. Moreover, $c_{\mathcal{I}} \in \mathcal{J}^\wedge$, thus by definition of $\psi_{\boldsymbol{\mathcal{U}}}$ there must exist an $S \in u$ such that $\mathcal{I} \subseteq S$. Hence $\mathcal{I} \in l^\uparrow \cap u^\downarrow = [\![l, u]\!]$. □

Example 6.4.10. *Considering the poset of coherent three-valued interpretations as worked out in Example 6.4.4 with $\Sigma^\wedge = \{p, q, c_{\{p,q\}}\}$. Let us take a closer look at some interesting elements in this poset.*

It is easily verified that $\psi_{\boldsymbol{\mathcal{L}}}(\{\}) = \{\}$, in fact, for every $l \in \boldsymbol{\mathcal{L}_f} \backslash \{\{p, q\}\}$, $\psi_{\boldsymbol{\mathcal{L}}}(l) = l$. The ALB $\{p, q\}$ is a little bit less straightforward, i.e., $\psi_{\boldsymbol{\mathcal{L}}}(\{p, q\}) = \{p, q, c_{\{p,q\}}\}$.

Similarly, for every $\{u\} \in \boldsymbol{\mathcal{U}_f}$, if $u \neq \{p, q\}$, then $\psi_{\boldsymbol{\mathcal{U}}}(\{u\}) = u$. On the other hand, then $\psi_{\boldsymbol{\mathcal{U}}}(\{\{p, q\}\}) = \{p, q, c_{\{p,q\}}\}$. In other words, the flowers that correspond to intervals are easily recognized in the set of coherent three-valued interpretations over $\Sigma^\wedge$, as long as we use the term $\{p, q, c_{\{p,q\}}\}$ to represent the two-valued interpretation $\{p, q\}$.

However, consider the AUB $\{\{p\}, \{q\}\}$, i.e., the only AUB in $\boldsymbol{\mathcal{U}_f}$ that does not result in interval-like flowers. Then $\psi_{\boldsymbol{\mathcal{U}}}(\{\{p\}, \{q\}\}) = \{p, q\}$. Hence, in a coherent three-valued interpretation over $\Sigma^\wedge$, the second component $\{p, q\}$ represents the AUB of flowers of interpretations where at most p or q is true, but not both. For example, $t^f(\{\}, \{p, q\}) = \{\{\}, \{p\}, \{q\}\}$. Furthermore, we see that $t^f(\{p\}, \{p, q\}) = \{\{p\}\} = \{p\}^\uparrow \cap \{\{p\}, \{q\}\}^\downarrow = [\![\{p\}, \{\{p\}, \{q\}\}]\!]$, showcasing how t^f essentially combines the information in the 'lower bound' $\{p\}$ and the 'upper bound' $\{p, q\}$ as expected.

In summary, we see that the poset of coherent three-valued interpretations over $\Sigma^\wedge$ equipped with the precision order over three-valued interpretations is strongly linked to the composition poset of flowers of two-valued interpretations over Σ. The goal is now to construct a new logic program $P^\wedge$ over $\Sigma^\wedge$ from P such that the three-valued immediate consequence operator $\Phi_{P^\wedge}^{SK}$ approximates T_P. We do this by constructing new rules to derive the conjunctional propositional atoms c_S building upon the rules for its conjuncts.

Definition 6.4.11. *Let P be a normal logic program*[3] *over a finite set of propositional atoms Σ. We define the set of* combined rule bodies B_S*, with $S \subseteq \Sigma$, inductively:*

[3] In a normal logic program the body of each rule corresponds to a conjunction of (positive or negative) literals. The ideas explained in this section can be generalized to logic programs with more advanced logics and with infinite sets of propositional atoms. However, such generalization requires careful consideration of how we combine rule bodies and how we replace conjunctions. This goes beyond the purpose of this example and we leave it to future work.

- $\psi \in B_S$ *if there exists a* $p \in \Sigma$ *such that* $S = \{p\}$ *and* $(p \leftarrow \psi) \in P$.
- $\psi \wedge \varphi \in B_S$ *if there exists a partition* $\{S_1, S_2\}$ *of* S *such that* $\psi \in B_{S_1}$ *and* $\varphi \in B_{S_2}$*, and there exists an interpretation* $\mathcal{I}$ *over* Σ *such that* $\mathcal{I} \models \psi \wedge \varphi$.

Informally, B_S consists of all satisfiable arguments that would cause the conjunction of S to become true.

Example 6.4.12 (Example 6.4.1, contd.)**.** *Remember the following simple program* P *over* $\Sigma = \{p, q\}$*:*

$$q \leftarrow p \wedge q.$$
$$p \leftarrow \neg q.$$
$$p \leftarrow \neg p.$$

As none of the rules that derive p *are compatible with the rule that derives* q*, we find the following combined rule bodies:*

- $B_{\{p\}} = \{\neg p, \neg q\}$,
- $B_{\{q\}} = \{p \wedge q\}$, *and*
- $B_{\{p,q\}} = \{\}$.

Assume that there was a fourth rule $q \leftarrow \neg p \wedge \neg q$*. Then we would have*

- $B_{\{p\}} = \{\neg p, \neg q\}$,
- $B_{\{q\}} = \{p \wedge q, \neg p \wedge \neg q\}$, *and*
- $B_{\{p,q\}} = \{\neg p \wedge \neg q \wedge \neg q, \neg p \wedge \neg q \wedge \neg p\}$.

Finally, we can construct a new normal logic program over $\Sigma^\wedge$ using the sets of combined rule bodies, by rephrasing them in terms of their dependencies on the truth of the conjunctional propositional atoms.

Definition 6.4.13. *Let* P *be a normal logic program over a finite set of propositional atoms* Σ*. The* conjunctional extension $P^\wedge$ *of* P *is the set of rules*

$$P^\wedge = \{(c_S \leftarrow replace(\psi).) \mid S \subseteq \Sigma \wedge \psi \in B_S\}$$

where $replace(\psi)$ *replaces the set* S *of positive literals in* ψ *by* c_S.

Definition 6.4.14. *Let* P *be a normal logic program over* Σ *and let* $P^\wedge$ *be the conjunctional extension of* P*. We define the conjunctional three-valued immediate*

consequence operator[4] $\Gamma_P : \mathcal{F} \to \mathcal{F}$ *on flowers of two-valued interpretations over* Σ *such that*

$$\Gamma_P(\boldsymbol{X}) = t^f(\Phi^{SK}_{P^\wedge}(\psi_{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}), \psi_{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{U}}})))$$

Proposition 6.4.15. *If* P *is a normal logic program over* Σ*, then* Γ_P *is an approximator on* $\mathcal{F}$*.*

Proof. We know $\Phi^{SK}_{P^\wedge}$ is an approximator of $T_{P^\wedge}$, i.e., it is monotonic and internal for three-valued interpretations over $\Sigma^\wedge$. Now we will also show that $\Phi^{SK}_{P^\wedge}$ is internal for the coherent subset.

Let $(\mathcal{I}^\wedge_2, \mathcal{J}^\wedge_2) = \Phi^{SK}_{P^\wedge}(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$, and let $S \subseteq \Sigma$. We see $c_S \in \mathcal{I}^\wedge_2$ iff there exists a $\psi \in B_S$ such that $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(replace(\psi)) = \mathbf{t}$. Since $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent, this means $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(\psi) = \mathbf{t}$. Furthermore, since $\psi \in B_S$, for every $s \in S$, there must exist a rule $(s \leftarrow \psi_s) \in P$ such that ψ_s is a sub-formula of ψ. Additionally, the conjunction of all these ψ_s is equivalent to ψ. Then by definition of $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}$ for conjunctions, $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(\psi) = \mathbf{t}$ iff $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(\psi_s) = \mathbf{t}$ for every $s \in S$. Moreover, note that for singletons $\{s\}$, $B_{\{s\}}$ is exactly the set of bodies of rules in P with s in the head, i.e., there exists a rule $(s \leftarrow \psi_s) \in P$ iff $\psi_s \in B_{\{s\}}$. Thus we find that $s \in \mathcal{I}^\wedge$ for every $s \in S$, hence $S \subseteq \mathcal{I}^\wedge$.

Similarly, assume $S' \subseteq S \subseteq \Sigma$ and $c_S \in \mathcal{J}^\wedge$. Then there must exist a $\psi \in B_S$ such that $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(replace(\psi)) \in \{\mathbf{t}, \mathbf{u}\}$. Since $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent, the truth value of c_S is lower than that of the conjunction $\bigwedge S$, hence $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(\psi) \in \{\mathbf{t}, \mathbf{u}\}$. Again, by definition of B_S, this entails that there must exist a rule $(s \leftarrow \psi_s) \in P$ for every $s \in S$ such that ψ_s is a sub-formula of ψ and ψ is equivalent to the conjunction of all ψ_s. Therefore, by definition of $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}$ for conjunctions, $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(\psi_s) \in \{\mathbf{t}, \mathbf{u}\}$. Furthermore, we know that ψ is satisfiable, i.e., there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \psi$. As a result, every conjunction of a subset of these ψ_s must be satisfiable as well. Hence, when we take the set of ψ_s for all $s \in S'$, the combined rule body ψ' of this set, i.e., the conjunction of the rule bodies, is satisfiable. As a result, $\psi' \in B_{S'}$. Furthermore, it is not hard to see that the truth value of ψ must be lower than that of ψ', hence $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}(\psi') \in \{\mathbf{t}, \mathbf{u}\}$. Consequently, $c_{S'} \in \mathcal{J}^\wedge$.

As a result, we find that Γ_P indeed maps flowers of two-valued interpretations to other flowers. Finally, since we know that $\psi_{\boldsymbol{\mathcal{L}}}$, $\psi_{\boldsymbol{\mathcal{U}}}$ t^f and Φ^{SK}_P are monotonic, we can conclude that Γ_P is monotonic as well, i.e., Γ_P is an approximator on $\mathcal{F}$. □

[4] We use Fitting's three-valued immediate consequence operator for the sake of simplicity. However, this can be generalized for a subset of the regular three-valued immediate consequence operators as defined in Chapter 4. In particular, it no longer suffices to have a regular three-valued truth-function. Instead, we additionally require that the truth-function coincides with Kleene's strong truth-evaluation for conjunctions in order to guarantee that the immediate consequence operator preserves coherence of the interpretations.

Proposition 6.4.16. *If P is a normal logic program over Σ with immediate consequence operator T_P, then Γ_P extends T_P.*

Proof. Γ_P extends T_P if for every (two-valued) interpretation $\mathcal{I} \in 2^\Sigma$, $\Gamma_P(\{\mathcal{I}\}) = \{T_P(\mathcal{I})\}$. It is not hard to see that $\psi_{\boldsymbol{\mathcal{L}}}(\{\mathcal{I}\}^{\boldsymbol{\mathcal{L}}}) = \mathcal{I}^\wedge = \psi_{\boldsymbol{\mathcal{U}}}(\{\mathcal{I}\}^{\boldsymbol{\mathcal{U}}})$ with $\mathcal{I}^\wedge = \{c_S \mid S \subseteq \mathcal{I}\}$. Moreover, it is not hard to see that since $(\mathcal{I}^\wedge, \mathcal{I}^\wedge)$ is coherent, $c_S \in \mathcal{I}^\wedge$ iff $S \subseteq \mathcal{I}^\wedge$, hence $S \subseteq \mathcal{I}^\wedge$ iff $S \subseteq \mathcal{I}$. Furthermore, for every $s \in \Sigma$, $B_S = \{\psi \mid (s \leftarrow \psi) \in P\}$. Then it is not hard to see that $T_P(\mathcal{I}) = T_{P^\wedge}(\mathcal{I}^\wedge) \cap \Sigma$. Since $\Phi^{SK}_{P^\wedge}$ extends $T_{P^\wedge}$, we know that $\Phi^{SK}_{P^\wedge}(\mathcal{I}^\wedge, \mathcal{I}^\wedge) = (T_{P^\wedge}(\mathcal{I}^\wedge), T_{P^\wedge}(\mathcal{I}^\wedge))$. Finally, by definition of t^f, we find that $t^f(\Phi^{SK}_{P^\wedge}(\mathcal{I}^\wedge, \mathcal{I}^\wedge)) = t^f(T_{P^\wedge}(\mathcal{I}^\wedge), T_{P^\wedge}(\mathcal{I}^\wedge)) = \{T_{P^\wedge}(\mathcal{I}^\wedge) \cap \Sigma\} = \{T_P(\mathcal{I})\}$. In other words, $\Gamma_P = (t^f \circ \Phi^{SK}_{P^\wedge} \circ t^\wedge)$ extends T_P. □

Example 6.4.17 (Example 6.4.1, contd.)**.** *Since none of the rules for p are compatible to the rule for q, we notice that $B_{\{p,q\}}$ is empty. As a result, we obtain the following simple logic program $P^\wedge$ over $\Sigma^\wedge$:*

$$
\begin{aligned}
q &\leftarrow c_{\{p,q\}}.\\
p &\leftarrow \neg p.\\
p &\leftarrow \neg q.
\end{aligned}
$$

Crucially, in $P^\wedge$, the first rule has changed, instead of having the conjunction $p \wedge q$ in the body, we now have the conjunctional atom $c_{\{p,q\}}$. However, there is no rule that can derive $c_{\{p,q\}}$, thus we can safely derive that $c_{\{p,q\}}$ is false, thus q is false and hence p must be true. This is indeed what happens if we consider the fastest monotonic induction of Γ_P:

$$
\begin{aligned}
\Gamma_P(2^\Sigma) &= t^f(\Phi^{SK}_{P^\wedge}(\psi_{\boldsymbol{\mathcal{L}}}((2^\Sigma)^{\boldsymbol{\mathcal{L}}}), \psi_{\boldsymbol{\mathcal{U}}}((2^\Sigma)^{\boldsymbol{\mathcal{U}}})))\\
&= t^f(\Phi^{SK}_{P^\wedge}(\psi_{\boldsymbol{\mathcal{L}}}(\{\}), \psi_{\boldsymbol{\mathcal{U}}}(\{\{p,q\}\})))\\
&= t^f(\Phi^{SK}_{P^\wedge}(\{\}, \{p,q,c_{\{p,q\}}\}))\\
&= t^f(\{\}, \{p,q\})\\
&= \{\{\}, \{p\}, \{q\}\}\\
&\quad\downarrow\\
\Gamma_P(\{\{\}, \{p\}, \{q\}\}) &= t^f(\Phi^{SK}_{P^\wedge}(\psi_{\boldsymbol{\mathcal{L}}}(\{\}), \psi_{\boldsymbol{\mathcal{U}}}(\{\{p\}, \{q\}\})))\\
&= t^f(\Phi^{SK}_{P^\wedge}(\{\}, \{p,q\}))\\
&= t^f(\{\}, \{p\})\\
&= \{\{\}, \{p\}\}\\
&\quad\downarrow\\
\Gamma_P(\{\{\}, \{p\}\}) &= t^f(\Phi^{SK}_{P^\wedge}(\psi_{\boldsymbol{\mathcal{L}}}(\{\}), \psi_{\boldsymbol{\mathcal{U}}}(\{\{p\}\})))
\end{aligned}
$$

$$
\begin{aligned}
&= t^f(\Phi^{SK}_{P^\wedge}(\{\}, \{p\})) \\
&= t^f(\{p\}, \{p\}) \\
&= \{\{p\}\}
\end{aligned}
$$

Notice that during a monotonic induction of Γ_P we can skip the intermediate conversions from flowers to three-valued interpretations over $\Sigma^\wedge$, and vice versa. Instead, we only need to convert the least precise flower in the beginning, then we follow a monotonic induction of $\Phi^{SK}_{P^\wedge}$. Only when this monotonic induction reaches a fixpoint, we convert it back to a flower. This way, the amount of overhead due to conversions is severely limited.

In this example, we see that finding the intended fixpoint with Γ_P demands a significantly less amount of computational power compared to using $\Phi^{SK}_{P^\wedge}$. In summary, even for simple logic programs P, Γ_P is a useful approximator which can be significantly stronger than the ultimate semantics in iAFT. The example above is fairly simple and results in a conjunctional program of which the immediate consequence operator is less complex rather than more complex. In general, the size of the conjunctional program may grow exponentially with respect to the number of rules in P. On the other hand, the resulting rules typically have much simpler bodies, which can significantly reduce the complexity of individual derivations. This makes the practical complexity difficult to anticipate. Nevertheless, conjunctional programs can become so large that they are no longer easy to reason about directly. Moreover, the construction of the conjunctional program itself may introduce a significant overhead. These observations motivate the following section, where we turn our attention to more condensed variants of conjunctional programs.

6.4.2 Condensed Conjunctional Immediate Consequence Operators

In this section, we revisit the auto-epistemic theory of Example 5.1.3, but now recast it as a logic program that reproduces the same effects. Compared to the simple logic program in Example 6.4.1, program is substantially larger than the original theory; moreover, there are more pairs of compatible rules. As a result, the conjunctional program grows significantly.

Example 6.4.18. *Recall the auto-epistemic theory in Example 5.1.3:*

$$\mathcal{T} = \left\{ \begin{array}{l} q \Leftrightarrow \neg K p. \\ r \Leftrightarrow \neg K q. \end{array} \right\}.$$

Our aim is now to construct a logic program that represents the same problem. In logic programming, we reason about interpretations. Since we denote interpretations $\mathcal{I}$ by the set of atoms that are true in $\mathcal{I}$, the exact space for logic programming corresponds to the power set lattice over a set of ground atoms. Therefore, the first thing we need to determine is this set. In particular, the set must be informative and diverse enough to model differentiating properties of different belief states. Notably, the semantic operator O of $\mathcal{T}$ only depends on the belief state and specifically on the truth values of Kp and Kq in this belief state. Therefore, all belief states with the same truth assignments for Kp and Kq are indistinguishable as inputs in the context of the semantic operator, thus we can group them together. Moreover, the operator maps all belief states to one out of four belief states which differ from each other in their evaluation of Kq, $K\neg q$, Kr or $K\neg r$. Hence, if we can differentiate between belief states based on their truth values of Kp, Kq, $K\neg q$, Kr and $K\neg r$, we should have sufficiently rich information to obtain the desired fixpoint. In this regard, we introduce the set of ground atoms $\{K_p, K_{\neg p}, K_q, K_{\neg q}, K_r, K_{\neg r}\}$ where we presume that if K_φ is true in an interpretation $\mathcal{I}$, then φ is known in the group of belief states represented by $\mathcal{I}$. Finally, we need special consideration of the inconsistent belief state. In this belief state everything is known, moreover, if two contradictory facts are derived to be true, we achieve the inconsistent belief state. To model this behavior, we need to add an additional ground atom K_i which represents inconsistent knowledge.

Now it is not hard to reformulate the problem as the following logic program P.

$$K_q \leftarrow \neg K_p. \qquad K_{\neg q} \leftarrow K_p.$$

$$K_r \leftarrow \neg K_q. \qquad K_{\neg r} \leftarrow K_q.$$

$$\forall x \in \{p, q, r\} : K_x \leftarrow K_i.$$

$$\forall x \in \{p, q, r\} : K_{\neg x} \leftarrow K_i.$$

$$\forall x \in \{p, q, r\} : K_i \leftarrow K_x \wedge K_{\neg x}.$$

Note that this corresponds to a slight variation of the original theory, since belief states that would originally collapse directly into the inconsistent belief state are now mapped through an intermediate interpretation first. However, given that the inconsistent belief state is generally considered unintuitive or unsatisfactory and that any rational agent would ignore mappings to the inconsistent belief state, as already argued in Section 6.2, this difference is not a major concern.

For this program, the conjunctional extension of Σ consists of 127 propositional atoms, while the conjunctional extension of P consists of 1023 rules, albeit with atomic bodies.

To improve the interpretability of conjunctional programs, we can curb their growth by restricting our attention to a carefully selected subset of the possible conjunctions.

Definition 6.4.19. *Let P be a normal logic program over a finite set of propositional atoms Σ and let $\Sigma \subseteq C \subseteq \Sigma^{\wedge}$. We define*

$$P^{\wedge}(C) = \{(c_S \leftarrow replace_C(\psi).) \mid c_S \in C \wedge \psi \in B_S\}$$

where $replace_C(\psi)$ replaces the set S of positive literals in ψ by a conjunction of a set $\{c_{S_i}\}_i$ of atoms $c_{S_i} \in C$ such that $S = \bigcup_i S_i$ and for every i, there is no set S' such that $S_i \subsetneq S' \subseteq S$ and $c_{S'} \in C$.

Example 6.4.20 (Example 6.4.18, contd.)**.** *For this program conjunctions only appear in the body of the rules of the form $K_i \leftarrow K_x \wedge K_{\neg x}$ with $x \in \{p, q, r\}$. Therefore, it is sensible to take $C = \Sigma \cup \{c_{\{K_x, K_{\neg x}\}} \mid x \in \{p, q, r\}\}$.*

We find the following sets of combined rule bodies:

$$B_{\{K_p, K_{\neg p}\}} = \{K_i \wedge K_i\}$$

$$B_{\{K_q, K_{\neg q}\}} = \{\neg K_p \wedge K_i, K_i \wedge K_p, K_i \wedge K_i\}$$

$$B_{\{K_r, K_{\neg r}\}} = \{\neg K_q \wedge K_i, K_i \wedge K_q, K_i \wedge K_i\}$$

As a result, we obtain the following program $P^{\wedge}(C)$:

$$K_q \leftarrow \neg K_p. \qquad K_{\neg q} \leftarrow K_p.$$

$$K_r \leftarrow \neg K_q. \qquad K_{\neg r} \leftarrow K_q.$$

$$\forall x \in \{p, q, r\} : K_x \leftarrow K_i.$$

$$\forall x \in \{p, q, r\} : K_{\neg x} \leftarrow K_i.$$

$$\forall x \in \{p, q, r\} : K_i \leftarrow c_{\{K_x, K_{\neg x}\}}.$$

$$c_{\{K_q, K_{\neg q}\}} \leftarrow K_i \wedge K_p.$$

$$c_{\{K_q, K_{\neg q}\}} \leftarrow K_i \wedge \neg K_p.$$

$$c_{\{K_r, K_{\neg r}\}} \leftarrow K_i \wedge K_q.$$

$$c_{\{K_r, K_{\neg r}\}} \leftarrow K_i \wedge \neg K_q.$$

$$\forall x \in \{p, q, r\} : c_{\{K_x, K_{\neg x}\}} \leftarrow K_i.$$

In this extended program we notice that new conjunctions of positive literals have appeared, in particular, we notice the conjunctions $K_i \wedge K_p$ and $K_i \wedge K_q$. Therefore, we might consider reiterating the procedure with $C' = C \cup \{c_{\{K_i, K_p\}}, c_{\{K_i, K_q\}}\}$. We refer the reader to the appendix for an iteratively obtained partial conjunctional program of this example. To reach a fixpoint, we require two additional iterations of the procedure which in total result in a C'' such that $|C''| = 12$, and a program with 33 newly added rules.

Such a program $P^{\wedge}(C)$ has no rules that could derive a conjunction c_S if $S \notin C$. As a result, the associated immediate consequence operator would derive falsity of c_S, oftentimes resulting in a non-coherent interpretation.

Definition 6.4.21. *Let Σ be a finite set of propositional atoms with conjunctional extension $\Sigma^{\wedge}$. Let C be a set of propositional atoms such that $\Sigma \subseteq C \subseteq \Sigma^{\wedge}$. We say a three-valued interpretation $(\mathcal{I}^{\wedge}, \mathcal{J}^{\wedge})$ over $\Sigma^{\wedge}$ is C-*coherent *if for every $c_S \in C$*

- *$c_S \in \mathcal{I}^{\wedge}$ iff $S \subseteq \mathcal{I}^{\wedge}$, and*
- *if $c_S \in \mathcal{J}^{\wedge}$, then for every $S' \subseteq S$ such that $c_{S'} \in C$, $c_{S'} \in \mathcal{J}^{\wedge}$.*

Despite that it is easily verified that $\Phi_{P^{\wedge}(C)}$ maps any coherent interpretation to a C-coherent interpretation, this does not suffice to ensure that the C-variant of our previously defined operator remains well-defined. Instead, we need to derive the truth-value of all $c_S \notin C$ implicitly, depending on the truth value of any explicitly derived smaller conjunctions. Specifically, we change our transformation t^f to reflect this implicit derivation.

Definition 6.4.22. *Let Σ be a finite set of propositional atoms and let $\Sigma^{\wedge}$ be the conjunctional extension of Σ. We define t^f_C as the function which maps a three-valued interpretation $(\mathcal{I}^{\wedge}, \mathcal{J}^{\wedge})$ over $\Sigma^{\wedge}$ to the set of all two-valued interpretations $\mathcal{I}$ over Σ satisfying*

- *$(\mathcal{I}^{\wedge} \cap \Sigma) \subseteq \mathcal{I}$, and*
- *for every $S \subseteq \mathcal{I}$, if $c_S \in C$, then $c_S \in \mathcal{J}^{\wedge}$.*

Finally, we are ready to define a syntactical approximator Γ^C_P analogously to Γ_P.

Definition 6.4.23. *Let P be a normal logic program over Σ. We define a partial conjunctional three-valued immediate consequence operator $\Gamma^C_P : \mathcal{F} \to \mathcal{F}$ on flowers of two-valued interpretations over Σ such that*

$$\Gamma^C_P(\boldsymbol{X}) = t^f_C(\Phi^{SK}_{P^{\wedge}(C)}(\psi_{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}), \psi_{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{U}}})))$$

Proposition 6.4.24. *Let P be a normal logic program over Σ. If $\Sigma \subseteq C \subseteq \Sigma^\wedge$, then Γ_P^C is an approximator.*

Proof. Completely analogous to the part of the proof for Proposition 6.4.15 where we show that $\Phi_{P^\wedge}^{SK}$ is internal for coherent three-valued interpretations, we can show that $\Phi_{P^\wedge(C)}^{SK}$ maps a coherent three-valued interpretation to a C-coherent interpretation.

Assume that $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is C-coherent. Since $\Sigma^\wedge$ is finite, it is trivial that $t_C^f(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is closed. Moreover, since $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is C-coherent, it is easily verified that $t_C^f(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is convex. Finally, clearly $\mathrm{glb}(t_C^f(\mathcal{I}^\wedge, \mathcal{J}^\wedge)) = \mathcal{I}^\wedge \cap \Sigma \in t_C^f(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$. Therefore $t_C^f(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is a flower as desired.

Finally, it suffices to show that t_C^f is monotonic to conclude that Γ_P^C is monotonic and thus an approximator. Let $(\mathcal{I}_1^\wedge, \mathcal{J}_1^\wedge)$ and $(\mathcal{I}_2^\wedge, \mathcal{J}_2^\wedge)$ be three-valued interpretations over $\Sigma^\wedge$. We know that $(\mathcal{I}_1^\wedge, \mathcal{J}_1^\wedge) \leq_p (\mathcal{I}_2^\wedge, \mathcal{J}_2^\wedge)$ iff $\mathcal{I}_1^\wedge \subseteq \mathcal{I}_2^\wedge \subseteq \mathcal{J}_2^\wedge \subseteq \mathcal{J}_1^\wedge$. Assume $\mathcal{I} \in t_C^f(\mathcal{I}_2^\wedge, \mathcal{J}_2^\wedge)$. Then $(\mathcal{I}_1^\wedge \cap \Sigma) \subseteq (\mathcal{I}_2^\wedge \cap \Sigma) \subseteq \mathcal{I}$. Moreover, for every $S \subseteq \mathcal{I}$, if $c_S \in C$, then $c_S \in \mathcal{J}_2^\wedge \subseteq \mathcal{J}_1^\wedge$. Therefore $\mathcal{I} \in t^f(\mathcal{I}_1^\wedge, \mathcal{J}_1^\wedge)$. Thus $t_C^f(\mathcal{I}_1^\wedge, \mathcal{J}_1^\wedge) \supseteq t_C^f(\mathcal{I}_2^\wedge, \mathcal{J}_2^\wedge)$. Hence, t_C^f is order-preserving. □

Note that the increase in size of the constructed program $P^\wedge(C)$ heavily depends on our choice of C. A bigger set C yields bigger programs $P^\wedge(C)$. On the other hand, the set C also indicates how much of the more accurate information provided by flowers is actually exploited by the approximator. In particular, we can only derive explicit information about the conjunctions in C, hence, the approximator always maps to flowers $\boldsymbol{X}$ such that $c_{\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}} \in C$. In other words, we cannot take advantage of the full space of flowers. In summary, this is another instance where we need to cautiously consider the balance between complexity and expressiveness.

Proposition 6.4.25. *Let P be a normal logic program over Σ. If $\Sigma \subseteq C_1 \subseteq C_2 \subseteq \Sigma^\wedge$, then $\Gamma_P^{C_1} \leq_p \Gamma_P^{C_2}$.*

Proof. Let $\boldsymbol{X} \in \mathcal{F}$ and let $(\mathcal{I}^\wedge, \mathcal{J}^\wedge) = (\psi_{\boldsymbol{\mathcal{L}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{L}}}), \psi_{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}^{\boldsymbol{\mathcal{U}}}))$. Let $(\mathcal{I}_1^\wedge, \mathcal{J}_1^\wedge) = \Phi_{P^\wedge(C_1)}^{SK}(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ and analogously, let $(\mathcal{I}_2^\wedge, \mathcal{J}_2^\wedge) = \Phi_{P^\wedge(C_2)}^{SK}(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$. Finally, we assume $\boldsymbol{Y}_1 = t_{C_1}^f(\mathcal{I}_1^\wedge, \mathcal{J}_1^\wedge)$ and $\boldsymbol{Y}_2 = t_{C_2}^f(\mathcal{I}_2^\wedge, \mathcal{J}_2^\wedge)$. Now we will show that if $\mathcal{I} \in \boldsymbol{Y}_2$, then $\mathcal{I} \in \boldsymbol{Y}_1$.

Clearly, for every $s \in \Sigma$, there is a bijection between rules $s \leftarrow \psi_1 \in P^\wedge(C_1)$ and rules $s \leftarrow \psi \in P$ such that $\psi_1 = replace_{C_1}(\psi)$. Similarly,there is a bijection between rules $s \leftarrow \psi_2 \in P^\wedge(C_2)$ and rules $s \leftarrow \psi \in P$ such that $\psi_2 = replace_{C_2}(\psi)$. Since $(\mathcal{I}^\wedge, \mathcal{J}^\wedge)$ is coherent, then for every $c_S \in \Sigma$, $c_S \in \mathcal{I}^\wedge$ iff $S \subseteq \mathcal{I}^\wedge$. Then it is not hard to see that $\mathcal{H}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}^{SK}(\psi_1) = \mathbf{t}$ iff $\mathcal{H}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}^{SK}(\psi) = \mathbf{t}$ iff $\mathcal{H}_{(\mathcal{I}^\wedge, \mathcal{J}^\wedge)}^{SK}(\psi_2) = \mathbf{t}$, by

definition of $replace_{C_1}$ and $replace_{C_2}$. As result, $(\mathcal{I}_1^\wedge \cap \Sigma) = (\mathcal{I}_2^\wedge \cap \Sigma)$. Therefore, if $\mathcal{I} \in \boldsymbol{Y}_2$, then $(\mathcal{I}_2^\wedge \cap \Sigma) \subseteq \mathcal{I}$, hence $(\mathcal{I}_1^\wedge \cap \Sigma) \subseteq \mathcal{I}$.

Additionally, if $\mathcal{I} \in \boldsymbol{Y}_2$, then $c_S \in \mathcal{J}_2^\wedge$ for every $S \subseteq \mathcal{I}$ such that $c_S \in C_2$. Since, $C_1 \subseteq C_2$, it also holds that $c_S \in \mathcal{J}_2^\wedge$ for every $S \subseteq \mathcal{I}$ such that $c_S \in C_1$. Moreover, if $c_S \in \mathcal{J}_2^\wedge$, then there must exist a rule $c_S \leftarrow \psi_2$ in $P^\wedge(C_2)$ such that $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge,\mathcal{J}^\wedge)}(\psi_2) \in \{\mathbf{t},\mathbf{u}\}$ and $\psi_2 = replace_{C_2}(\psi)$ for some $\psi \in B_S$. Let us now look at $\psi_1 = replace_{C_1}(\psi)$, then by definition of $replace_{C_1}$ and $replace_{C_2}$, for every (positively) occurring conjunct c_{S_1} in ψ_1, there must exist a (positively) occurring conjunct c_{S_2} in ψ_2 where $S_2 \supseteq S_1$. Moreover, by definition of $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge,\mathcal{J}^\wedge)}$ for conjunctions, if $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge,\mathcal{J}^\wedge)}(\psi_2) \in \{\mathbf{t},\mathbf{u}\}$, then for every (positively) occurring propositional atom c_{S_2} in ψ_2, $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge,\mathcal{J}^\wedge)}(c_{S_2}) \in \{\mathbf{t},\mathbf{u}\}$, i.e., $c_{S_2} \in \mathcal{J}^\wedge$. Hence, since $(\mathcal{I}^\wedge,\mathcal{J}^\wedge)$ is coherent, for every (positively) occurring propositional atom c_{S_1}, we must have that $c_{S_1} \in \mathcal{J}^\wedge$ as well. Consequently, since the negatively occurring propositional atoms remain unchanged, we find that if $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge,\mathcal{J}^\wedge)}(\psi_2) \in \{\mathbf{t},\mathbf{u}\}$, then $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge,\mathcal{J}^\wedge)}(\psi_1) \in \{\mathbf{t},\mathbf{u}\}$, by definition of $\mathcal{H}^{SK}_{(\mathcal{I}^\wedge,\mathcal{J}^\wedge)}$ for conjunctions. In other words, if $c_S \in \mathcal{J}_2^\wedge$, then $c_S \in \mathcal{J}_1^\wedge$.

In summary, if $\mathcal{I} \in \boldsymbol{Y}_2$, then $\mathcal{I} \in \boldsymbol{Y}_1$, hence $\Gamma_P^{C_1}(\boldsymbol{X}) = \boldsymbol{Y}_1 \supseteq \boldsymbol{Y}_2 = \Gamma_P^{C_2}(\boldsymbol{X})$, i.e., $\Gamma_P^{C_1}(\boldsymbol{X}) \leq_p \Gamma_P^{C_2}(\boldsymbol{X})$. Since $\boldsymbol{X}$ was chosen arbitrarily, this entails that $\Gamma_P^{C_1} \leq_p \Gamma_P^{C_2}$. □

Corollary 6.4.26. *Let P be a normal logic program over a finite set of propositional atoms Σ. If $\Sigma \subseteq C \subseteq \Sigma^\wedge$, then $\Phi_P^f \leq_p \Gamma_P^C \leq_p \Gamma_P$, where Φ_P^f is the natural extension of Φ_P to the space of flowers.*

Proof. It is easily verified that $\Gamma_P^\Sigma = \Phi_P^f$ and $\Gamma_P^{\Sigma^\wedge} = \Gamma_P$. The proposition then trivially follows from Proposition 6.4.25. □

Corollary 6.4.27. *Let P be a normal logic program over a finite set of propositional atoms Σ. If $\Sigma \subseteq C \subseteq \Sigma^\wedge$, then Γ_P^C extends T_P.*

Proof. This trivially follows from the fact that Φ_P^f and Γ_P both extend T_P, and $\Phi_P^f \leq_p \Gamma_P^C \leq_p \Gamma_P$ □

Example 6.4.28 (Example 6.4.18, contd.)**.** *Using the previously worked out normal logic program $P^\wedge(C)$ with $C = \Sigma \cup \{c_{\{K_x,K_{\neg x}\}} \mid x \in \{p,q,r\}\}$, we notice that, while significantly less complex than Γ_P, the approximator Γ_P^C is already sufficiently refined to derive the intended fixpoint. To show this, let us look at the fastest monotonic induction of its stable revision operator St.*

Iteration 1

$$\Gamma_P^C(2^\Sigma) = t_C^f(\Phi^{SK}_{P^\wedge(C)}(\{\}, \Sigma^\wedge)) = t_C^f(\{\}, C) = 2^\Sigma$$

Thus, again $St^1(2^\Sigma) = \mathrm{glb}(2^\Sigma) = \{\}$. *In other words, as expected, we still cannot refine the lower bound during the first iteration. However,*

$$\begin{aligned}
&\Gamma_P^C(\{\{\}\}) = t_C^f(\Phi^{SK}_{P^\wedge(C)}(\{\}, \{\})) \\
&\quad = t_C^f(\{K_q, K_r\}, \{K_q, K_r\}) = \{\{K_q, K_r\}\} \\
&\qquad \downarrow \\
&\Gamma_P^C(\{K_q, K_r\}^\downarrow) = t_C^f(\Phi^{SK}_{P^\wedge(C)}(\{\}, \{K_q, K_r, c_{\{K_q,K_r\}}\})) \\
&\quad = t_C^f(\{K_q\}, \{K_q, K_r, K_{\neg r}\}) \\
&\quad = \{\{K_q\}, \{K_q, K_r\}, \{K_q, K_{\neg r}\}\} \\
&\qquad \downarrow \\
&\Gamma_P^C(\{\{K_q\}, \{K_q, K_r\}, \{K_q, K_{\neg r}\}\}^\downarrow) \\
&\quad = t_C^f(\Phi^{SK}_{P^\wedge(C)}(\{\}, \{K_q, K_r, K_{\neg r}, c_{\{K_q,K_r\}}, c_{\{K_q,K_{\neg r}\}}\})) \\
&\quad = t_C^f(\{K_q\}, \{K_q, K_r, K_{\neg r}\}) \\
&\quad = \{\{K_q\}, \{K_q, K_r\}, \{K_q, K_{\neg r}\}\}
\end{aligned}$$

Hence, we are able to refine the upper bound, i.e.,

$$\begin{aligned}
St^2(2^\Sigma) &= Max(\{\{K_q\}, \{K_q, K_r\}, \{K_q, K_{\neg r}\}\}) \\
&= \{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}
\end{aligned}$$

Consequently, $St(2^\Sigma) = \{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}^\downarrow$.

Iteration 2

$$\begin{aligned}
&\Gamma_P^C(\{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}^\downarrow) = \{\{K_q\}, \{K_q, K_r\}, \{K_q, K_{\neg r}\}\} \\
&\downarrow \mathrm{glb}(\{\{K_q\}, \{K_q, K_r\}, \{K_q, K_{\neg r}\}\}) = \{K_q\} \\
&\Gamma_P^C(\{\{K_q\}, \{K_q, K_r\}, \{K_q, K_{\neg r}\}\}) \\
&\quad = t_C^f(\Phi^{SK}_{P^\wedge(C)}(\{K_q\}, \{K_q, K_r, K_{\neg r}, c_{\{K_q,K_r\}}, c_{\{K_q,K_{\neg r}\}}\})) \\
&\quad = t_C^f(\{K_q, K_{\neg r}\}, \{K_q, K_{\neg r}\}) \\
&\quad = \{\{K_q, K_{\neg r}\}\}
\end{aligned}$$

Since $\mathrm{glb}(\{\{K_q, K_{\neg r}\}\}) = \{K_q, K_{\neg r}\}$, *we find* $St^1(\{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}^{\downarrow}) = \{K_q, K_{\neg r}\}$. *Moreover, since* $(2^{\Sigma})^{\mathcal{L}} = \{\} = (\{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}^{\downarrow})^{\mathcal{L}}$,

$$St^2(\{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}^{\downarrow}) = St^2(2^{\Sigma}) = \{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}.$$

In summary, $St(\{\{K_q, K_r\}, \{K_q, K_{\neg r}\}\}^{\downarrow}) = \{\{K_q, K_{\neg r}\}\}$.

We see that this derivation of the intended fixpoint completely mirrors the fastest monotonic induction of the ultimate approximator of T_P. *Hence, the approximator* Γ_P^C *is a syntactically defined alternative to* U_{T_P} *with a reasonable complexity that does not compromise on expressiveness.*

In conclusion, to avoid the potential blow-up in size, more condensed variants of the conjunctional program can be used, often without losing the intended semantics. In this way, we use the additional information of flowers while keeping the conjunctional program interpretable, ensuring that the framework remains practical for larger or more intricate examples.

6.5 Approximating across Non-Monotonic Reasoning Formalisms

In the preceding section, we modeled a slight variation of Example 5.1.3 using an alternative approach, namely by formulating it as a logic program. At first sight, the program seems to capture the same problem. However, while they are both forms of constructive knowledge, these two logics are inherently very different in nature. This naturally raises the question: how can we, in general, verify whether two modelings indeed represent the same problem or knowledge? In this section, we show how gAFT provides concrete tools to address this question. Still, in its current form, the framework lacks certain results needed to fully support our findings in this example, highlighting an important direction for future research.

Notably, when comparing different representations of a problem, it does not suffice to check whether they arrive at equivalent solutions. Importantly, it does not suffice to merely check whether they produce equivalent outcomes. For instance, consider the logic program consisting of the fact p.' versus the program with the rule $p \leftarrow \neg q$'. Both programs yield the same well-founded model, namely p. Yet, the justifications for why p holds are fundamentally different: in the first case p is accepted outright as a fact, while in the second case its truth hinges on the absence of q. With gAFT, comparing different representations of one problem becomes streamlined and formalized. Specifically, we need to show (1) that the approximation space associated with the alternative approach

is in fact a (preferably non-degenerate) approximation space of the original exact space related to the problem, and (2) that any approximator of the alternative semantic operator is an approximator of the original semantic operator.

We start by showing that the approximation space in iAFT for the logic program is an approximation space of the original exact space of belief states, i.e., $\langle 2^\Sigma \times 2^\Sigma, \leq_p \rangle$ approximates the space of all belief states over $\{p, q, r\}$, with $\Sigma = \{K_p, K_{\neg p}, K_q, K_{\neg q}, K_r, K_{\neg r}, K_i\}$. We know from Example 5.4.21 that $\langle 2^\Sigma \times 2^\Sigma, \leq_p \rangle$ is an approximation space with $\boldsymbol{\mathcal{L}} = \langle 2^\Sigma, \subseteq \rangle = \boldsymbol{\mathcal{U}}$.

Proposition 6.5.1. *The approximation space $\langle 2^\Sigma \times 2^\Sigma, \leq_p \rangle$ approximates the space $\boldsymbol{\mathcal{E}}$ of all belief states over $\{p, q, r\}$ with $\preceq_{\boldsymbol{\mathcal{E}}}$ which extends $\preceq$ and $\leq$ such that for all $\mathcal{I} \in 2^\Sigma$ and $\mathcal{X} \in \boldsymbol{\mathcal{E}}$:*

- $\mathcal{I} \preceq_{\boldsymbol{\mathcal{E}}} \mathcal{X}$ *if* $K_i \notin \mathcal{I} \wedge \forall \varphi \in \{p, q, r, \neg p, \neg q, \neg r\} : K_\varphi \in \mathcal{I} \implies \mathcal{X} \models K\varphi$
- $\mathcal{X} \preceq_{\boldsymbol{\mathcal{E}}} \mathcal{I}$ *if* $\forall \varphi \in \{p, q, r, \neg p, \neg q, \neg r\} : \mathcal{X} \models K\varphi \implies K_\varphi \in \mathcal{I}$

where $\mathcal{X} \models K\varphi$ if φ is true in every interpretation $\mathcal{J} \in \mathcal{X}$.

Proof. At first glance, the relation we defined on $2^\Sigma \cup \boldsymbol{\mathcal{E}}$ is not a proper order, since distinct elements coming from different component posets may in fact correspond to the same factual value. Strictly speaking, this means the relation lacks anti-symmetry. However, this redundancy is purely representational, the multiple copies of the same value arise from the way we combine the posets, not from a genuine semantic difference. In this sense, $\preceq_{\boldsymbol{\mathcal{E}}}$ is a partial order on some $2^\Sigma \cup \boldsymbol{\mathcal{E}}'$ with $\boldsymbol{\mathcal{E}}'$ isomorphic to $\boldsymbol{\mathcal{E}}$. From its definition as an extension of $\preceq$ and $\leq$, it trivially follows that this order is in fact a global approximant order. Moreover, by defining $\sim$ such that for every $l \in \boldsymbol{\mathcal{L}}$, $u \in \boldsymbol{\mathcal{U}}$ and $x \in \boldsymbol{\mathcal{E}}$: $[\![l, u]\!] \sim x$ iff $l \preceq_{\boldsymbol{\mathcal{E}}} x \preceq_{\boldsymbol{\mathcal{E}}} u$, 1 in Definition 5.6.11 is trivially satisfied. Finally, conditions 2 and 3 trivially follow from the fact that the exact space $\boldsymbol{\mathcal{E}}$ and the approximation space $\langle 2^\Sigma \times 2^\Sigma, \leq_p \rangle$ are finite, hence all chains in these spaces are finite as well. As a result, the least upper bound of any chain is the maximal element of said chain. □

Note that this approximation space has a lot of $\leq_p$-maximal approximants which are not exact. On the one hand, we have a lot of approximants of the form (x, x) which are empty, i.e., if $K_i \in x$ or if it partially represents inconsistent knowledge, e.g., $K_p \in x$ and $K_{\neg p} \in x$, but $K_q \notin x$. On the other hand, plenty of the $\leq_p$-maximal approximants will approximate a group of belief states that assign the same truth values to Kp, $K\neg p$, Kq, $K\neg q$, Kr and $K\neg r$. For example assume $x = \{K_p\}$, then (x, x) approximates the belief state $\{\{p, q, r\}, \{p, \neg q, \neg r\}\}$ as well as the belief state $\{\{p, q, r\}, \{p, \neg q, \neg r\}, \{p, q, \neg r\}\}$, among others. To be precise, only 21 of the 128 $\leq_p$-maximal approximants are exact. Specifically, we have the case where $x =$

$\{K_p, K_{\neg p}, K_q, K_{\neg q}, K_r, K_{\neg r}\}$ such that (x, x) approximates exactly the inconsistent belief state, and any approximant (x, x) such that $|x| \geq 2$ and $x \subseteq \{K_{p^*}, K_{q^*}, K_{r^*}\}$ where $p^* \in \{p, \neg p\}$, $q^* \in \{q, \neg q\}$ and $r^* \in \{r, \neg r\}$. Additionally, for every $\leq_p$-maximal exact approximant (x, x), we see there is another exact approximant $(x, x \cup \{K_i\})$ which is not $\leq_p$-maximal. In other words, this approach shows us a practical example of a good and sensible approximation space where the relation between the exact space, the $\leq_p$-maximal approximants and the exact approximants is significantly severed, resulting in degenerate approximants.

Proposition 6.5.2. *Let $A \in Appr(\langle 2^\Sigma \times 2^\Sigma, \leq_p \rangle)$. If A approximates T_P, then A approximates $O_\mathcal{T}$, when the latter is treated as a partial operator which ignores mappings to the inconsistent belief state.*

Proof. Let us define the mapping $f : \boldsymbol{\mathcal{E}} \to 2^\Sigma$ such that f maps any belief state $\mathcal{X}$ to the interpretation $\mathcal{I}$ consisting of all ground atoms K_φ such that φ is known in $\mathcal{X}$. Then we can show that for any belief state $\mathcal{X}$, $(f \circ O_\mathcal{T})(\mathcal{X}) = (T_P \circ f)(\mathcal{X})$, provided that $O_\mathcal{T})(\mathcal{X})$ is not the inconsistent belief state. This can be verified on a case-by-case basis, but we omit these details here for the sake of brevity.

Furthermore, it is not hard to see that, for every $x, y \in 2^\Sigma$ and $\mathcal{X}$, $(x, y) \sim \mathcal{X}$ iff $x \subseteq f(\mathcal{X}) \subseteq y$, i.e., $f(\mathcal{X}) \in [x, y]$, by definition of $\sim$. Consequently, if A approximates T_P and $(x, y) \sim \mathcal{X}$, then $f(O_\mathcal{T}(\mathcal{X})) = T_P(f(\mathcal{X})) \in A(x, y)$. Therefore $A(x, y) \sim O_\mathcal{T}(\mathcal{X})$. Thus A must approximate $O_\mathcal{T}$. □

As a result, the iAFT-ultimate approximator $U^i_{T_P}$ of T_P is an approximators of $O_\mathcal{T}$, and is are Fitting's three-valued immediate consequence operator Φ^{SK}_P. Note moreover, that despite the approximation space not being non-degenerate with respect to $\boldsymbol{\mathcal{E}}$, these approximators are seemingly sufficiently well-behaved, i.e., their well-founded fixpoints approximate the intended solution, as shown by the following example.

Example 6.5.3 (Example 6.4.18, contd.)**.** *Let us compute the well-founded fixpoint of $U^i_{T_P}$ using the fastest monotonic induction of its stable revision operator. Notice that*

$$T_P(\{\}) = \{K_q, K_r\}$$

$$T_P(\{K_q, K_r\}) = \{K_q, K_{\neg r}\}$$

$$T_P(\{K_p, K_q\}) = \{K_{\neg q}, K_{\neg r}\}$$

$$T_P(\{K_p, K_{\neg p}, K_i\}) = \{K_p, K_{\neg p}, K_q, K_{\neg q}, K_r, K_{\neg r}, K_i\} = \Sigma$$

*Therefore, in **Iteration 1** we have*

$$U^i_{T_P}(\{\}, \Sigma) = (\{\}, \Sigma)$$

Thus, $St^1(\{\}, \Sigma)) = \{\}$. *Moreover,*

$$\begin{array}{l} U^i_{T_P}(\{\}, \{\}) = (\{K_q, K_r\}, \{K_q, K_r\}) \\ \downarrow \\ U^i_{T_P}(\{\}, \{K_q, K_r\}) = (\{K_q\}, \{K_q, K_r, K_{\neg r}\}) \\ \downarrow \\ U^i_{T_P}(\{\}, \{K_q, K_r, K_{\neg r}\}) = (\{K_q\}, \{K_q, K_r, K_{\neg r}, K_i\}) \\ \downarrow \\ U^i_{T_P}(\{\}, \{K_q, K_r, K_{\neg r}, K_i\}) = (\{K_q\}, \Sigma) \\ \downarrow \\ U^i_{T_P}(\{\}, \Sigma) = (\{\}, \Sigma) \end{array}$$

As a result, $St^2(\{\}, \Sigma) = \Sigma$. *Hence, after one iteration we already see that* $WF(U^i_{T_P}) = (\{\}, \Sigma)$.

Since $U^i_{T_P}$ is the most precise approximator of T_P using iAFT, we can safely conclude that intervals are not precise enough to find the intended well-founded semantics, just as when we applied iAFT to the original AEL-theory. Now one might wonder what happens if we instead use fAFT to solve this logic program. Again, we see that the corresponding approximation space is also an approximation space of $\boldsymbol{\mathcal{E}}$.

Proposition 6.5.4. *The approximation space* $\langle \mathcal{F}_{2^\Sigma}, \supseteq \rangle$ *approximates the space* $\boldsymbol{\mathcal{E}}$ *of all belief states over* $\{p, q, r\}$ *with* $\preceq_{\boldsymbol{\mathcal{E}}}$ *extending* $\preceq$ *and* $\leq$ *such that for all* $\mathcal{I} \in \boldsymbol{\mathcal{L}_f}$, $u \in \boldsymbol{\mathcal{U}_f}$ *and* $\mathcal{X} \in \boldsymbol{\mathcal{E}}$*:*

- $\mathcal{I} \preceq_{\boldsymbol{\mathcal{E}}} \mathcal{X}$ *if* $K_i \notin \mathcal{I} \wedge \forall \varphi \in \{p, q, r, \neg p, \neg q, \neg r\} : K_\varphi \in \mathcal{I} \implies \mathcal{X} \models K\varphi$
- $\mathcal{X} \preceq_{\boldsymbol{\mathcal{E}}} u$ *if there exists an* $\mathcal{I} \in u$*:* $\forall \varphi \in \{p, q, r, \neg p, \neg q, \neg r\} : \mathcal{X} \models K\varphi \implies K_\varphi \in \mathcal{I}$

where $\mathcal{X} \models K\varphi$ *if* φ *is true in every interpretation* $\mathcal{I} \in \mathcal{X}$.

Proof. This is analogous to the proof of Proposition 6.5.1 when we use the definitions of $\langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ as defined in Example 5.4.2 for this specific instantiation. □

Since $\leq_p$-maximal approximants coincide for iAFT and fAFT, the discussion of exact approximants in the latter approximation space is analogous to that of iAFT. Furthermore, it is easy to see that, as for iAFT, every approximator of T_P in fAFT will be an approximator that interpolates $O_\mathcal{T}$ too.

Proposition 6.5.5. *Let $A \in Appr(\mathcal{F}_{\mathbf{2}^\Sigma})$. If A approximates T_P, then A approximates $O_\mathcal{T}$, when the latter is treated as a partial operator which ignores mappings to the inconsistent belief state.*

Proof. Completely analogous to Proposition 6.5.2. □

As a result, the ultimate operator $U^f_{T_P}$ on $\mathcal{F}_{2^\Sigma}$ of T_P is an approximator of $O_\mathcal{T}$ and so are the newly introduced conjunctional immediate consequence operators. In Example 6.4.28 we have shown that the program indeed finds the expected belief state with fAFT. Again, these natural approximators are seemingly well-behaved enough to establish some form of correctness.

In conclusion, this section demonstrates how different representations of the same problem, situated in distinct logics or contexts, can be meaningfully compared within our framework. At the same time, it exposes a gap: while for non-degenerate approximation spaces we are guaranteed results consistent with earlier work, such guarantees do not automatically extend to other spaces. Nevertheless, our use of natural approximators here suggests that a similar correctness property may still hold more generally, a conjecture that calls for further investigation.

6.6 Conclusion

While Chapter 5 is primarily concerned with the theoretical foundations and formal properties involved in extending AFT to more general settings, this chapter touches upon the practical implications and benefits of this generalization. The most immediate advantages of gAFT become clear in situations where iAFT, the original instantiation, is unable to produce meaningful results, yet other instances, such as fAFT, are capable of successfully computing the intended fixpoint. One concrete example of this phenomenon is found in Example 5.1.3, where we consider a simple but illustrative auto-epistemic theory. In this example, fAFT is able to refine approximations and ultimately identify the correct fixpoint, whereas iAFT fails to make any progress at all. In fact, we believe that fAFT will prove sufficiently expressive for a significant subset of the auto-epistemic theories. This suggests an important increase in expressiveness that gAFT brings to the table.

When we approach this example from the point of view of Normal Logic Programming, it further demonstrates that there also exist logic programs for which iAFT does not provide sufficient expressive power, whereas fAFT succeeds in finding the intended model. This shows that fAFT can also be a useful extension within the context of LP. Typically, we start investigating AFT semantics by using the ultimate approximator, as, theoretically, it requires little effort to define and understand it. That said, while

the ultimate operator serves as an excellent theoretical tool, providing a clear upper bound on the expressiveness of an gAFT instance, it is often less well-suited for practical applications. In the specific context of logic programming with iAFT, Fitting's three-valued immediate consequence operator Φ_P^{SK} is often preferred, as it offers a more tractable and intuitive way to reason about program behavior. Motivated by this, we explored alternative approximators that capture the intuitions of Fitting's operator but operate on flowers instead of intervals, i.e., the conjunctional immediate consequence operator Γ_P and its more condensed variants. Notably, the operator Γ_P for fAFT successfully extends the intuitiveness and compositional explainability of the interval-based approach to the richer approximation space of flowers. By explicitly handling conjunctions, it leverages the additional expressive power of fAFT while preserving interpretability for ground atoms. However, recognizing that full exploitation of this space may lead to computational overhead and reduced transparency for conjunctions, we also introduced a family of intermediate operators between Φ_P^{SK} and Γ_P. These condensed operators form a spectrum balancing accuracy, efficiency, and interpretability, offering practical alternatives tailored to specific reasoning tasks. Together, they demonstrate that the benefits of fAFT can be realized in a controlled manner, aligning theoretical richness with computational and explanatory feasibility.

Furthermore, this chapter turns its attention to the improved flexibility of gAFT compared to iAFT with respect to the exact space. A key limitation of iAFT is its strict requirement that the exact space must form a complete lattice. If this condition is not met, iAFT cannot even be applied. By contrast, gAFT greatly expands the range of settings in which approximation fixpoint theory can be used by allowing for different kinds of underlying structures. For instance, if the exact space forms a bounded-complete cpo rather than a complete lattice, fAFT can be applied directly and without any additional overhead, making it both natural and efficient in such contexts. Alternatively, one can extend a bounded-complete cpo to a complete lattice by adding suitable top elements, thereby enabling the use of an interval-based version of AFT. Importantly, this extension can itself be understood and verified as yet another instance of gAFT, ensuring conceptual coherence within the framework. Nevertheless, fAFT is often the more natural fit for such exact spaces, as it preserves the structure rather than artificially extending it. This means that it can provide more refined upper bounds and richer approximations that better capture the inherent characteristics of the problem domain. However, as illustrated by Example 6.3.15, even when the framework seems to be a natural fit for the problem domain, this does not automatically guarantee that every problem can be solved.

The practical potential implications of gAFT extend well beyond simply increasing expressiveness. One of the key strengths of this generalization lies in its ability to abstract away from a fixed approximation space, thereby providing a unifying framework that can be applied across multiple domains of non-monotonic reasoning. This abstraction not only introduces greater flexibility in how semantic operators are

analyzed and applied, but it also deepens our overall understanding of both AFT and the broader principles of non-monotonic reasoning. By working at a more general level, gAFT gives us the tools to formally compare and relate semantics that were previously treated as separate or domain-specific. More concretely, it allows us to investigate when and how an approximation space defined for one semantic operator can be used for another. It enables us to establish connections between different research areas, exact spaces, and operators. In this way, gAFT acts as both a practical tool and a conceptual bridge between logics.

Besides showcasing practical improvements of fAFT and gAFT, this chapter also highlights some of the remaining gaps in the research. The most pressing one concerns the restriction to non-degenerate approximation spaces in order to guarantee correctness of the well-founded fixpoint. Indeed, for spaces with degenerate approximants it is often possible to construct approximators of a semantic operator O that display counter-intuitive or undesirable behavior. However, when we restrict our attention to natural approximators, their behavior appears far more well-behaved, suggesting that the correctness results may extend more broadly than currently proven.

It is important to emphasize that this chapter does not provide an exhaustive list of the practical impact of gAFT, nor does it explore in depth all the aspects it touches upon. Many questions remain open, and further research is required to verify our hypotheses and fully understand the extent of these practical implications. The discussion here is intended mainly to give the reader a preliminary sense of the possibilities that gAFT opens up, highlighting the potential for future work in this area.

Chapter 7

Conclusion

7.1 Our Contributions

This thesis set out to develop a deeper theoretical understanding of formalizations of constructive knowledge by focusing on the common thread that unites its many manifestations: the construction process. Our first objective was to analyze, generalize, and formalize this notion in a systematic way. To that end, Chapter 3 examined one of the most common yet powerful forms of constructive knowledge, namely monotonic constructive definitions. While these have been studied extensively across different domains, prior work has remained fragmented. By unifying these perspectives under a single framework, we revealed that a constructive definition can be understood in general as a *semantic operator* acting on an ordered *construction space*, with the construction process itself formalized as a *monotonic induction* on that space.

A central contribution of this work is the observation that, despite originating from seemingly disparate domains, all such definitions share the same structural components. Their distinguishing features arise not from the construction process itself, but from the underlying construction space. Making these differences explicit enables two advances: first, it lays the foundation for logics that support diverse kinds of constructive definitions; second, it provides a clear criterion for understanding which results in one domain may transfer to others and which are domain-specific.

Whereas monotonic operators can be captured through monotonic inductions, non-monotonic operators require a more refined approach. Here, the Approximation Fixpoint Theory (AFT) and its associated well-founded inductions provide a principled framework. It is by now well established that AFT successfully models the core principles of non-monotonic induction. Moreover, logic programs have been argued

to constitute a formalization of inductive definitions. In this spirit, Pelov [67] showed how AFT can systematically extend logic programs with constructs such as aggregates. Yet, despite this, consensus within the LP and ASP communities on how to handle aggregates remains elusive, leading to recurring rediscoveries of variations of Pelov's ideas. A key contribution of this thesis, developed in Chapter 4, is to address precisely this gap. By reformulating AFT in terms that are more directly aligned with recent research regarding this topic, the link becomes clear and the theory becomes significantly more accessible. Furthermore, by demonstrating its application to concrete KR formalisms, both as a semantics for non-monotonic operators and as a methodological guide for extending such formalisms, the work showcases the practical relevance of AFT. As such, Chapter 4 addresses both the second and the third goal posed for this thesis.

In Chapter 5 we further advanced the second central goal of this thesis by extending AFT itself. Through a series of motivating examples across different domains, it became clear that existing versions of AFT were too restrictive. Our contribution lies in showing that by relaxing the notions of approximation space and approximant, these challenges can be systematically overcome. For this, we analyzed the core requirements of an approximation space. Within this context, we identified the most general formulation that still yields a conservative extension of AFT. As such, the proposed generalized framework covers most of the already existing consistent AFT-versions. This work contributes in two complementary ways. First, it significantly broadens the expressiveness and applicability of AFT across diverse domains. Second, it deepens the theoretical understanding of the principles that make approximation-based reasoning work. By isolating the essential properties of approximation spaces, we now have clear criteria to determine whether a space qualifies directly or requires additional constructs such as virtual elements. Importantly, our extension preserves both the fixpoint-based semantics and the constructive approach via well-founded inductions, maintaining the computational benefits and explanatory clarity that are key strengths of AFT.

Finally, in Chapter 6 we addressed the third research aim: demonstrating the applicability of our generalized version of AFT (gAFT) within the broader context of knowledge representation. First, we showcased concrete instantiations of gAFT, such as fAFT and relaxed iAFT, analyzing both their strengths and limitations. We explored their use in auto-epistemic logic, extended the role of iAFT in logic programming to fAFT, and investigated their applicability to non-monotonic recursive definitions as well as more generally to operators on bounded-complete cpos. Second, we highlighted how these generalizations enable AFT to capture a wider range of constructive processes in KR than was previously possible. More broadly, our results suggest that gAFT has the potential to elevate the unifying power of AFT: moving beyond the unification of semantics within a single domain towards a framework capable of comparing and connecting semantics across different areas of KR. This represents an important step

towards a deeper, more systematic understanding of constructive knowledge across logic-based formalisms.

7.2 Future Directions

A Logic with Constructive Definitions Looking ahead, an important goal is to translate the theoretical insights from Chapter 3 into the design of formal logics that can accommodate diverse types of constructive definitions. Since we have identified the two key components of such definitions, i.e., the semantic operator (derived from a set of rules) and the construction space, a natural next step is to develop a syntax that allows these components to be automatically extracted. While deriving the semantic operator is likely to be relatively straightforward, determining the appropriate construction space poses a greater challenge. Indeed, it is precisely the construction space that differentiates the various kinds of constructive definitions, and thus understanding how to formally capture it, will be crucial for building expressive and unifying knowledge representation languages.

Lower-regularity and ASP By reformulating the AFT framework in terms familiar to that audience, we discovered that not all semantics follow the same foundational view of logic programs as constructive definitions. Nonetheless, the strong structural connections indicate that many mathematical relations remain to be explored. For example, we hypothesize that if a *TSR* guarantees $\leq_t$-minimal stable models, then it must exhibit lower-regularity for convex rule bodies. Similarly, since all considered *TSRs* were lower-regular for anti-monotonic rule bodies, we conjecture that this property may be necessary to ensure that the semantics of a logic program remain invariant under strong equivalence transformations. Investigating and validating such hypotheses constitutes a promising line of future research, deepening the theoretical understanding of AFT and its role in unifying different non-monotonic semantics.

Extending AFT with Partial Operators Whereas the increased expressiveness of fAFT compared to iAFT in the context of auto-epistemic logic (AEL) is undeniable, as exemplified by Example 5.1.3, we also observed that certain well-motivated AEL examples remain unresolved. We attribute this limitation to the fact that the associated operators in these cases are not consistency-preserving. From a conceptual standpoint, however, it is reasonable to assume that a rational agent would disregard mappings that lead to the inconsistent belief state, since such a state is, by definition, deemed impossible. Interpreted this way, the operator in question is best understood as a partial operator. This observation points towards an important extension: the need to apply

AFT not only to total operators but also to partial operators, thereby broadening its applicability to more realistic reasoning scenarios.

A Correctness Property for Degenerate Approximants While we successfully established that for non-degenerate approximation spaces certain guarantees can be given regarding the consistency of models across different approximation spaces and approximators, we have not yet formulated corresponding guarantees for approximation spaces with degenerate approximants. Nevertheless, as illustrated by Example 5.6.23 and Section 6.5, there do exist meaningful approximation spaces that yield empty—and thus degenerate—approximants. Moreover, as demonstrated in Example 6.3.15, degenerate approximants may in fact be indispensable for providing semantics to more intricate forms of non-monotonic recursive definitions. While we recognize that not every approximator on such spaces will behave well or align with our expectations, we hypothesize that most natural approximators will. We believe that by introducing suitable constraints or regulations on the behavior of approximators in the presence of empty approximants, it may be possible to completely eliminate the undesired side-effects associated with degeneracy, while still retaining their expressive benefits.

gAFT and Other Versions of AFT Finally, we have established that several existing variants of AFT, such as iAFT [30] and hAFT [22, 12], can be viewed as instances of the generalized AFT framework. In contrast, we did not attempt to incorporate dAFT [50], motivated by the fundamentally different nature of non-deterministic programs, which seems to require distinct definitions of both the stable revision operator and the associated approximators. Nevertheless, we see potential in gAFT for this application, specifically in the extended space of non-empty convex sets. In particular, convex sets naturally allow lower bounds to consist of multiple minimal elements, analogous to the multiple upper bounds seen in flowers. While multiple upper bounds correspond intuitively to conjunctions, we argue that multiple lower bounds correspond to disjunctions. As illustrated by Example 5.6.23, however, this space requires the addition of virtual elements. We expect that the need for alternative definitions of the stable revision operator and approximators is closely related to the presence of these virtual elements. Exploring this connection further constitutes a promising avenue for future research.

Appendix A

gAFT and Necessary Conditions

In Chapter 5 we have shown that under reasonable assumptions, the generalized notion of approximation spaces as described in Section 5.4 yields a theory that retains the essential characteristics of the original versions of AFT. We have also concluded that iAFT and hAFT are instances of this generalized framework. Moreover, in Section 6.1 we show that fAFT is an instance of the framework as well. In other words, the introduced framework encapsulates many of the existing versions of AFT while also extending it with new, more complex and more expressive instances. On the other hand, the discussion in Example 5.6.23 clearly demonstrates that the space of non-empty closed and convex sets does not directly yield an approximation space according to our definition. Nevertheless, it is natural to think of non-empty closed and convex sets as approximations of exact elements. This raises the question whether there are even more general possible definitions for an approximation space leading to the same results. This appendix serves to prove that the conditions imposed on the approximation space are indeed the minimal ones, they are necessary to satisfy all requirements formulated in Section 5.3. Note again that the goal was to conservatively extend standard versions of AFT, hence the specified requirements need to be viewed in this context. Therefore, it is not unexpected that our framework does not directly capture dAFT which poses a different stable revision operator along side its revised approximation space. Consequently, this work should not be read as 'the most general version of AFT', but it can be understood as the most general overarching framework combining all possible consistent approximation spaces that virtually adopt the original stable revision operator.

The following sections consider every condition posed in the definitions of the

decomposition spaces and of the approximation space. For every condition we proof its necessity by contradiction. We assume that the condition does not hold, and construct a context for which at least one of the requirements is not satisfied. In particular, they constitute the satisfaction of Requirement 1 and Requirement 5. In this section, we do not discuss the conditions in the definition of a composition poset. Nevertheless, they are also necessary. However, they are direct consequences of the requirements and as such, their necessity has already been argued at the time of their introduction.

A.1 Decomposition Spaces

Most of the properties associated to the decomposition spaces are immediate consequences of the requirements proposed in Section 5.3. In particular, Requirement 1 claims that the approximation space must be a cpo. Consequently, it must have a least element $\bot_{\mathcal{A}}$. From Requirement 2, it then follows that $\bot_{\mathcal{A}}^{\mathcal{L}} \in \mathcal{L}$ and $\bot_{\mathcal{A}}^{\mathcal{U}} \in \mathcal{U}$ exists. Moreover, since for every other approximant $X \in \mathcal{A}$, $\bot_{\mathcal{A}} \leq_p X$, by Requirement 3, there is a partial order $\preceq$ over $\mathcal{L} \cup \mathcal{U}$ such that $\bot_{\mathcal{A}}^{\mathcal{L}} \preceq X^{\mathcal{L}} \preceq X^{\mathcal{U}} \preceq \bot_{\mathcal{A}}^{\mathcal{U}}$. Since $\mathcal{L}$ and $\mathcal{U}$ are the result of taking the decompositions of the approximants in $\mathcal{A}$, this entails that for every $b \in \mathcal{L} \cup \mathcal{U}$, $\bot_{\mathcal{A}}^{\mathcal{L}} \preceq b \preceq \bot_{\mathcal{A}}^{\mathcal{U}}$. In summary, $\bot_{\mathcal{A}}^{\mathcal{L}}$ must be the least element $\bot$ in $\mathcal{L} \cup \mathcal{U}$ and similarly, $\bot_{\mathcal{A}}^{\mathcal{U}}$ must be the greatest element $\top$. Hence, our assumption that $\mathcal{L}$ and $\mathcal{U}$ are sets and that $\preceq$ is a partial order over $\mathcal{L} \cup \mathcal{U}$ such that $\langle \mathcal{L} \cup \mathcal{U}, \preceq \rangle$ has a least element $\bot \in \mathcal{L}$ and greatest element $\top \in \mathcal{U}$ is a direct consequence of the requirements, and therefore, a necessary one. In contrast, showing that $\mathcal{L}$ is a cpo or that $\mathcal{U}$ is a co-chain-complete cpo demands a little more effort.

Proposition A.1.1. *Let $\langle \mathcal{A}, \leq_p \rangle$ be a composition poset[1] of $\mathcal{L}$ and $\mathcal{U}$. If the induced subposet $\mathcal{L} = \langle \mathcal{L}, \preceq \rangle$ is not a cpo, then there exists a monotonic operator $A : \mathcal{A} \to \mathcal{A}$ such that $St^1([\![\bot, \top]\!])$ is not well-defined.*

Proof. Proof by construction. We have already argued that $\mathcal{L}$ should have a least element. Hence, if $\mathcal{L}$ is not a cpo, then there must exist a non-empty chain $C \subseteq \mathcal{L}$, such that C does not have a least upper bound in $\mathcal{L}$. Clearly, the chain must be infinite, otherwise the maximal element in the chain would be the least upper bound. By Proposition 2.1.17, we know that C has a well-ordered cofinal subset $\{l_i\}_{i<\lambda}$ (with λ a (possibly transfinite) ordinal). Moreover, by Proposition 2.1.18, we know that $\{l_i\}_{i<\lambda}$ cannot have a least upper bound. Assume that $\{l_i\}_{i<\lambda}$ is ordered by $\preceq$, i.e., for every

[1] Note that a composition poset is strictly speaking only defined in case $\mathcal{L}$ and $\mathcal{U}$ are a lower and upper decomposition space, respectively. However, this definition is trivially extended to cases where $\mathcal{L}$ and $\mathcal{U}$ satisfy all conditions in the decomposition spaces except that $\mathcal{L}$ is not required to be a cpo and $\mathcal{U}$ is not necessarily a co-chain-complete cpo. This generalization is what is intended here and in the following two propositions.

$i, j < \lambda$, if $i \leq j$, then $l_i \preceq l_j$. Take the operator A such that

$$A(\boldsymbol{X}) = \begin{cases} [\![l_i, \top]\!] & \text{if } l_i \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \text{ and } l_j \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \text{ for every } j < i \\ \boldsymbol{X} & \text{if for every } i < \lambda,\ l_i \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \end{cases}$$

It is easy to see that A is monotonic and well-defined. If there exists an $l \in \{l_i\}_{i<\lambda}$ such that $l \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$, then the set $\{l \in \{l_i\}_{i<\lambda} \mid l \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}}\}$ is not empty. Hence by definition of a well-ordered set, it must have a least element l_*. Consequently, $A(\boldsymbol{X})$ is defined as $[\![l_*, \top]\!]$. Hence, A is always uniquely defined, i.e., it is well-defined. Moreover, by definition of a composition poset $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$ for $\boldsymbol{X} \leq_p \boldsymbol{Y}$. Therefore, for every $i < \lambda$, $l_i \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$ entails $l_i \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$, and $l_i \not\preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$ entails $l_i \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}}$. The monotonicity of A can be derived straightforwardly from these observations.

Now we can show that for any $l \in \boldsymbol{\mathcal{L}}$, l is a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\top}$ iff l is an upper bound of $\{l_i\}_{i<\lambda}$. Clearly, if l is an upper bound of $\{l_i\}_{i<\lambda}$, then the approximant $[\![l, \top]\!]$ is an instance of the second case in the definition of A. Hence, $A([\![l, \top]\!]) = [\![l, \top]\!]$. Therefore, $A^{\boldsymbol{\mathcal{L}}}_{\top}(l) = A^{\boldsymbol{\mathcal{L}}}([\![l, \top]\!]) = [\![l, \top]\!]^{\boldsymbol{\mathcal{L}}} = l$. Consequently, l is a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\top}$.

If l is not an upper bound of $\{l_i\}_{i<\lambda}$, then there must exist at least one $i < \lambda$ such that $l_i \not\preceq l$. Consequently, the first rule of the definition applies and $A([\![l, \top]\!]) = [\![l_*, \top]\!]$ where l_* is obtained as before. Hence, $A([\![l, \top]\!]) = ([\![l_*, \top]\!])$. Thus, $A^{\boldsymbol{\mathcal{L}}}_{\top}(l) = A^{\boldsymbol{\mathcal{L}}}([\![l, \top]\!]) = [\![l_*, \top]\!]^{\boldsymbol{\mathcal{L}}} = l_*$. Since $l_* \not\preceq l$, we have $l_* \neq l$, therefore l is not a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\top}$.

In summary, for any $l \in \boldsymbol{\mathcal{L}}$, l is a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\top}$ iff l is an upper bound of $\{l_i\}_{i<\lambda}$. Since by assumption, $\{l_i\}_{i<\lambda}$ does not have a least upper bound, $A^{\boldsymbol{\mathcal{L}}}_{\top}$ does not have a least fixpoint. Consequently, $St^1([\![\bot, \top]\!]) = \mathrm{lfp}(A^{\boldsymbol{\mathcal{L}}}_{\top})$ is not well-defined. □

Consequently, if $\boldsymbol{\mathcal{L}}$ is not a cpo, then it is impossible to guarantee that the stable revision operator is well-defined for the least precise approximant. However, since the least approximant $[\![\bot, \top]\!]$ is the least upper bound of the empty set, this entails that Requirement 5 is not satisfied. In a similar fashion, Requirement 5 implies that $\boldsymbol{\mathcal{U}}$ must be a cpo as a natural consequence of the following proposition.

Proposition A.1.2. *Let $\langle \boldsymbol{\mathcal{A}}, \leq_p \rangle$ be a composition poset of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$. If $\boldsymbol{\mathcal{U}} = \langle \mathcal{U}, \preceq \rangle$ is not a cpo, then there exists an operator $A : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{A}}$ such that $St^2([\![\bot, \top]\!])$ is not well-defined.*

Proof. Proof by construction. If the $\langle \boldsymbol{\mathcal{U}}, \preceq \rangle$ is not a cpo, then either $\boldsymbol{\mathcal{U}}$ does not have a least element or there exists a non-empty chain $C \subseteq \boldsymbol{\mathcal{U}}$, such that $\mathrm{lub}_{\boldsymbol{\mathcal{U}}}(C)$ does not exist.

For the first case, take the operator A such that $A(\boldsymbol{X}) = \boldsymbol{X}$. Clearly, this operator is monotonic and well-defined. Moreover, every $u \in \boldsymbol{\mathcal{U}}$ is a fixpoint of $A_{\perp}^{\boldsymbol{\mathcal{U}}}$. Since $\boldsymbol{\mathcal{U}}$ does not have a least element, $A_{\perp}^{\boldsymbol{\mathcal{U}}}$ does not have a least fixpoint. Therefore, $St^2([\![\perp, \top]\!])$ is not well-defined.

In the second case, the proof is completely analogous to that of Proposition A.1.1. The considered chain C must be infinitely large, otherwise the maximal element in the chain would be the least upper bound. By Proposition 2.1.17, we know that C has a well-ordered cofinal subset $\{u_i\}_{i<\lambda}$ (with λ a (possibly transfinite) ordinal). Moreover, by Proposition 2.1.18, we know that $\{u_i\}_{i<\lambda}$ does not have a least upper bound. Assume that $\{u_i\}_{i<\lambda}$ is ordered by $\preceq$, i.e., for every $i, j < \lambda$, if $i \leq j$, then $l_i \preceq l_j$. Take the operator A such that

$$A(\boldsymbol{X}) = \begin{cases} [\![\perp, u_i]\!] & \text{if } u_i \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \text{ and } u_j \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \text{ for every } j < i \\ \boldsymbol{X} & \text{if for every } i < \lambda,\ u_i \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \end{cases}$$

Analogous to the operator defined in the proof of Proposition A.1.1, it is easily verified that A is also monotonic and well-defined. Moreover, analogously to what was done in the proof of Proposition A.1.1 we can show that for any $u \in \boldsymbol{\mathcal{U}}$, u is a fixpoint of $A_{\perp}^{\boldsymbol{\mathcal{U}}}$ iff u is an upper bound of $\{u_i\}_{i<\lambda}$. Since by assumption, $\{u_i\}_{i<\lambda}$ does not have a least upper bound, $A_{\perp}^{\boldsymbol{\mathcal{U}}}$ does not have a least fixpoint. Consequently, $St^2([\![\perp, \top]\!]) = \mathrm{lfp}(A_{\perp}^{\boldsymbol{\mathcal{U}}})$ is not well-defined. □

Moreover, the following proposition implies that Requirement 1 entails that $\boldsymbol{\mathcal{U}}$ must be co-chain-complete. In fact, a similar reasoning additionally supports our claim that $\boldsymbol{\mathcal{L}}$ must be a cpo, i.e., for $\boldsymbol{\mathcal{A}}$ to be a cpo, $\boldsymbol{\mathcal{L}}$ needs to be a cpo as well.

Proposition A.1.3. *Let $\langle \boldsymbol{\mathcal{A}}, \leq_p \rangle$ be a composition poset of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$. If $\boldsymbol{\mathcal{U}} = \langle \mathcal{U}, \preceq \rangle$ is not a co-cpo, then $\boldsymbol{\mathcal{A}}$ is not a cpo.*

Proof. We already argued that $\boldsymbol{\mathcal{U}}$ must have a greatest element $\top$. Therefore, there must exist a non-empty chain $C \subseteq \boldsymbol{\mathcal{U}}$, such that $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$ does not exists.

Let us assume that $\boldsymbol{\mathcal{A}}$ is a cpo. For any chain $C \subseteq \boldsymbol{\mathcal{U}}$, we have that $C' = \{[\![\perp, u]\!] \mid u \in C\}$ is a chain in $\boldsymbol{\mathcal{A}}$ by anti-monotonicity of $[\![.,.]\!]$ in the second component. Since by assumption $\boldsymbol{\mathcal{A}}$ is a cpo, $\boldsymbol{X} = \mathrm{lub}_{\boldsymbol{\mathcal{A}}}(C')$ exists. Take $u = \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}$. Clearly, by anti-monotonicity of $.^{\boldsymbol{\mathcal{U}}}$, u must be a lower bound of C.

Moreover, for every lower bound u' of C, we have $\boldsymbol{Y} \leq_p [\![\perp, u']\!]$ for every $\boldsymbol{Y} \in C'$, again by anti-monotonicity of $[\![.,.]\!]$ in the second component. Therefore, $\boldsymbol{X} \leq_p [\![\perp, u']\!]$ by definition of the least upper bound. Consequently, by anti-monotonicity of $.^{\boldsymbol{\mathcal{U}}}$,

$$u' = [\![\perp, u']\!]^{\boldsymbol{\mathcal{U}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} = u.$$

In summary, u must be greatest lower bound of C. This contradicts our earlier assumption that C does not have a greatest lower bound $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$. □

In summary, Requirements 1 and 5 entail that $\boldsymbol{\mathcal{L}}$ must be a cpo while $\boldsymbol{\mathcal{U}}$ must be a co-chain-complete cpo. As a result, together with the aforementioned immediate consequences of the different requirements, we obtain that within the context of the proposed requirements, $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ are a lower and upper decomposition space, respectively.

A.2 Interlattice Properties

Finally, all that remains to be shown is the necessity of the interlattice properties. Specifically, we will prove that they are implied by Requirement 5. The proofs are similar to that of Proposition A.1.1, we assume that the interlattice properties are not satisfied and construct an approximator such that the stable revision operator is not well-defined and internal for any cpo containing the least (precise) approximant.

Proposition A.2.1. *Let $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ be a lower and upper decomposition space, respectively. Let $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$ be a composition poset of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ under $\preceq$. If the tuple $\langle \mathcal{L}, \mathcal{U}, \mathcal{A}, \preceq, \leq_p \rangle$ does not satisfy the Chain Interlattice Lub Property, then there exists an operator $A : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{A}}$ such that $St(St([\![\bot, \top]\!]))$ is not well-defined.*

Proof. Proof by construction. If the Chain Interlattice Lub Property is not satisfied, then there exists an $u \in \boldsymbol{\mathcal{U}}$ and a chain $C \subseteq [\bot, u]_{\boldsymbol{\mathcal{L}}}$, such that $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) \not\preceq u$. Clearly, C cannot be empty, otherwise $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C) = \bot$ and for every $u \in \boldsymbol{\mathcal{U}}$, $\bot \preceq u$. By Proposition 2.1.17, we know that C has a well-ordered co-final subset $\{l_i\}_{i<\lambda}$ (with λ a (possibly transfinite) ordinal). Moreover, by Proposition 2.1.18, we know that $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(\{l_i\}_{i<\lambda}) = \mathrm{lub}_{\boldsymbol{\mathcal{L}}}(C)$, therefore $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(\{l_i\}_{i<\lambda}) \not\preceq u$. Hence, the chain $\{l_i\}_{i<\lambda}$ must be infinite, otherwise the maximal element in the chain would be the least upper bound. Assume that $\{l_i\}_{i<\lambda}$ is ordered by $\preceq$, i.e., for every $i, j < \lambda$, if $i \leq j$, then $l_i \preceq l_j$. Take the operator A such that

$$A(\boldsymbol{X}) = \begin{cases} [\![l_i, u]\!] & \text{if } [\![l_i, u]\!]^{\boldsymbol{\mathcal{L}}} \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}},\ \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \preceq u \text{ and for every } j < i,\ [\![l_j, u]\!]^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \\ \boldsymbol{X} & \text{if } \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \preceq u \text{ and for every } i < \lambda,\ [\![l_i, u]\!]^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \\ [\![\bot, u]\!] & \text{if } \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \not\preceq u \end{cases}$$

It is again easily verified that A is well-defined. It is not hard to check that the conditions in each of the cases are mutually exclusive. Furthermore, by a similar reasoning as for Proposition A.1.1, we obtain that there will always be one applicable rule. Moreover, we can show that A is monotonic by a case-by-case proof. Note that there are three

possible scenarios combining two approximants $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that $\boldsymbol{X} \leq_p \boldsymbol{Y}$. In the first scenario $\boldsymbol{X}$ is an instance of the third case, in the second both belong to the second case, and in the third $\boldsymbol{X}$ is an instance of the first case while $\boldsymbol{Y}$ is not an instance of the third case. Proving monotonicity in the first or second scenario is trivial. For the third scenario we again use the fact that $\boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \preceq \boldsymbol{Y}^{\boldsymbol{\mathcal{L}}}$ if $\boldsymbol{X} \leq_p \boldsymbol{Y}$.

Now, let us assume that $St(\llbracket \bot, \top \rrbracket)$ is well-defined. If this is not the case, then $St(St(\llbracket \bot, \top \rrbracket))$ cannot be well-defined either. Clearly, for every $\boldsymbol{X} \in \boldsymbol{\mathcal{A}}$, $\llbracket \bot, u \rrbracket \leq_p A(\boldsymbol{X})$, therefore, $A^{\boldsymbol{\mathcal{U}}}(\boldsymbol{X}) \preceq \llbracket \bot, u \rrbracket^{\boldsymbol{\mathcal{U}}} = u$. Consequently, $St^2(\llbracket \bot, \top \rrbracket) \preceq u$, since $St^2(\llbracket \bot, \top \rrbracket)$ is a fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{\bot}$. In summary, if $u' = St(\llbracket \bot, \top \rrbracket)^{\boldsymbol{\mathcal{U}}}$, then $u' \preceq u$.

Again, similar to what was done in the proof of Proposition A.1.1 we can now show that if $l \in \boldsymbol{\mathcal{L}}$ is a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{u'}$, then l must be an upper bound of $\{l_i\}_{i<\lambda}$.

If l is not an upper bound of $\{l_i\}_{i<\lambda}$, then there must exist at least one $i < \lambda$ such that $l_i \not\preceq l$. Let l_* be the least element of the set $\{l' \in \{l_i\}_{i<\lambda} \mid l' \not\preceq l\}$. Then $A^{\boldsymbol{\mathcal{L}}}_{u'}(l) = \llbracket l_*, u \rrbracket^{\boldsymbol{\mathcal{L}}}$. Moreover, $l_* \preceq \llbracket l_*, u \rrbracket^{\boldsymbol{\mathcal{L}}}$ and $l_* \not\preceq l$, hence $A^{\boldsymbol{\mathcal{L}}}_{u'}(l) = \llbracket l_*, u \rrbracket^{\boldsymbol{\mathcal{L}}} \neq l$. Thus l is not a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{u'}$.

In summary for any $l \in \boldsymbol{\mathcal{L}}$, if l is a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{u}$, then l is an upper bound of $\{l_i\}_{i<\lambda}$. Since by assumption, $\mathrm{lub}_{\boldsymbol{\mathcal{L}}}(\{l_i\}_{i<\lambda}) \notin [\bot, u]_{\boldsymbol{\mathcal{L}}}$, $A^{\boldsymbol{\mathcal{L}}}_{u'}$ has no upper bounds (and therefore no fixpoints) in $[\bot, u]_{\boldsymbol{\mathcal{L}}}$, including in $[\bot, u']_{\boldsymbol{\mathcal{L}}}$. Finally, since $A^{\boldsymbol{\mathcal{L}}}_{u'}$ is only well-defined for the domain $[\bot, u']_{\boldsymbol{\mathcal{L}}}$, $A^{\boldsymbol{\mathcal{L}}}_{u'}$ has no fixpoints, and therefore also no least fixpoint. Consequently, $St^1(St(\llbracket \bot, \top \rrbracket)) = \mathrm{lfp}(A^{\boldsymbol{\mathcal{L}}}_{u'})$ is not well-defined. Therefore, $St(St(\llbracket \bot, \top \rrbracket))$ is not well-defined. □

As a result, this implies that if the Chain Interlattice Lub Property is not satisfied, it cannot be guaranteed that there exists a cpo that coincides with the approximation space for the least upper bounds, and for which the stable revision operator is monotonic, well-defined and internal. In particular, such a cpo needs to contain the least approximant $\llbracket \bot, \top \rrbracket$, since this is the least upper bound of the empty set in $\boldsymbol{\mathcal{A}}$. Moreover, for it to be internal, $St(\llbracket \bot, \top \rrbracket)$ must be in this cpo as well, and thus, the stable revision operator should be well-defined for $St(\llbracket \bot, \top \rrbracket)$, which, as we have shown above, cannot be guaranteed. In summary, if the Chain Interlattice Lub Property is not satisfied, then there are approximators for which Requirement 5 is not satisfied either. Similarly, we prove that Requirement 5 implies the Interlattice Lb Property. First we show the implication of an analogous property to the Chain Interlattice Lub Property for the greatest lower bound. Afterwards, we use this property to also prove the entailment of the more general Interlattice Lb Property.

Proposition A.2.2. *Let $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ be a lower and upper decomposition space, respectively. Let $\boldsymbol{\mathcal{A}} = \langle \mathcal{A}, \leq_p \rangle$ be a composition poset of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ under $\preceq$. If for some $l \in \boldsymbol{\mathcal{L}}$ and some chain $C \subseteq [l, \top]_{\boldsymbol{\mathcal{U}}}$, $l \not\preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$, then there exists an operator $A : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{A}}$ such that $St(St(\llbracket \bot, \top \rrbracket))$ is not well-defined.*

Proof. Clearly, the chain C cannot be empty, otherwise, $\top = \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$ and for every $l \in \boldsymbol{\mathcal{L}}$, $l \preceq \top$. Let us consider $\boldsymbol{\mathcal{U}}^{\succeq} = \langle \mathcal{U}, \succeq \rangle$, i.e., $\boldsymbol{\mathcal{U}}$ with an inverted order. Clearly, since the order is just inverted, C must also be a chain in $\boldsymbol{\mathcal{U}}^{\succeq}$. Moreover, since $\boldsymbol{\mathcal{U}}$ is a co-chain-complete cpo, $\boldsymbol{\mathcal{U}}^{\succeq}$ must be a co-chain-complete cpo as well. As a result, the least upper bound and greatest lower bound of C exists for both.

By Proposition 2.1.17, we know that $C \subseteq \boldsymbol{\mathcal{U}}^{\succeq}$ has a well-ordered co-final subset $\{u_i\}_{i<\lambda}$ (with λ a (possibly transfinite) ordinal). Moreover, by Proposition 2.1.18, we know that $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(\{u_i\}_{i<\lambda}) = \mathrm{lub}_{\boldsymbol{\mathcal{U}}^{\succeq}}(\{u_i\}_{i<\lambda}) = \mathrm{lub}_{\boldsymbol{\mathcal{U}}^{\succeq}}(C) = \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$, therefore $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(\{u_i\}_{i<\lambda}) \not\preceq u$. Moreover, the chain $\{u_i\}_{i<\lambda}$ must be infinite, otherwise the least element in the chain would be the greatest lower bound. Assume that $\{u_i\}_{i<\lambda}$ is ordered by $\succeq$, i.e., for every $i, j < \lambda$, if $i \leq j$, then $l_i \succeq l_j$. Take the operator A such that

$$A(\boldsymbol{X}) = \begin{cases} [\![l, u_i]\!] & \text{if } l \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}},\ [\![l, u_i]\!]^{\boldsymbol{\mathcal{U}}} \not\succeq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \text{ and for every } j < i,\ [\![l, u_j]\!]^{\boldsymbol{\mathcal{U}}} \succeq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \\ \boldsymbol{X} & \text{if } l \preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \text{ and for every } i < \lambda,\ [\![l, u_i]\!]^{\boldsymbol{\mathcal{L}}} \succeq \boldsymbol{X}^{\boldsymbol{\mathcal{U}}} \\ [\![l, \top]\!] & \text{if } l \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \end{cases}$$

Analogous to the operator defined in the previous proof, it is easily verified that A is monotonic and well-defined.

Let us assume that $St([\![\bot, \top]\!])$ is well-defined. Otherwise, $St(St([\![\bot, \top]\!]))$ cannot be well-defined either. Clearly, for every $\boldsymbol{X}$, $[\![l, \top]\!] \leq_p A(\boldsymbol{X})$, hence $l = [\![l, \top]\!]^{\boldsymbol{\mathcal{L}}} \preceq A^{\boldsymbol{\mathcal{L}}}(\boldsymbol{X})$. Therefore, $l \preceq St^1([\![\bot, \top]\!])$, since $St^1([\![\bot, \top]\!])$ is a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\top}$. Hence, if $l' = St([\![\bot, \top]\!])^{\boldsymbol{\mathcal{L}}}$, then $l \preceq l'$.

Analogous to the previous proof of Proposition A.1.1 we now show that every fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{l'}$ is a lower bound of $\{u_i\}_{i<\lambda}$ in $\boldsymbol{\mathcal{U}}$. If u is not a lower bound of $\{u_i\}_{i<\lambda}$, then there must exist at least one $i < \lambda$ such that $u_i \not\succeq u$. Let u_* be the least element of the set $\{u' \in \{u_i\}_{i<\lambda} \mid u \not\preceq u'\}$. Then $A^{\boldsymbol{\mathcal{U}}}_{l'}(u) = [\![l, u_*]\!]^{\boldsymbol{\mathcal{U}}}$. Moreover, $u \not\preceq u_*$ and $[\![l, u_*]\!]^{\boldsymbol{\mathcal{U}}} \preceq u_*$, hence $u \neq [\![l, u_*]\!]^{\boldsymbol{\mathcal{U}}}$. Thus, u is not a fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{l'}$.

In summary, for any $u \in \boldsymbol{\mathcal{U}}$, if u is a fixpoint of $A^{\boldsymbol{\mathcal{U}}}_{l'}$, then u is a lower bound of $\{u_i\}_{i<\lambda}$. Since by assumption, $\mathrm{glb}_{\boldsymbol{\mathcal{L}}}(\{l_i\}_{i<\lambda}) \notin [l, \top]_{\boldsymbol{\mathcal{U}}}$, $\{l_i\}_{i<\lambda}$ has no lower bounds (and therefore $A^{\boldsymbol{\mathcal{U}}}_{l'}$ has no fixpoints) in $[l, \top]_{\boldsymbol{\mathcal{U}}}$, including in $[l', \top]_{\boldsymbol{\mathcal{U}}}$. Finally, since $A^{\boldsymbol{\mathcal{U}}}_{l'}$ is only well-defined for the domain $[l', \top]_{\boldsymbol{\mathcal{U}}}$, $A^{\boldsymbol{\mathcal{U}}}_{l'}$ has no fixpoints, and therefore also no least fixpoint. Consequently, $St^1(St([\![\bot, \top]\!])) = \mathrm{lfp}(A^{\boldsymbol{\mathcal{U}}}_{l'})$ is not well-defined. Therefore $St(St([\![\bot, \top]\!]))$ is not well-defined. □

Proposition A.2.3. *Let $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ be a lower and upper decomposition space, respectively. Let $\langle \mathcal{A}, \leq_p \rangle$ be a composition poset of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ under $\preceq$. If $\langle \boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{U}}, \mathcal{A}, \preceq, \leq_p \rangle$ does not satisfy the Interlattice Lb Property, then there exists an operator $A : \boldsymbol{\mathcal{A}} \to \boldsymbol{\mathcal{A}}$ such that $St(St([\![\bot, \top]\!]))$ is not well-defined.*

Proof. If the Interlattice Lb Property is not satisfied, then there must exist a set $S \subseteq \boldsymbol{\mathcal{U}}$ such that for every $u \in S$, $l \preceq u$, however, there is no $u' \in \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(S)$ such that $l \preceq u'$. If S is a chain, then the property follows immediately from Proposition A.2.2.

Now let us assume that the Interlattice Lb Property holds for every chain (including the empty chain), hence S is not a chain and thus non-empty. Take the operator A such that

$$A(\boldsymbol{X}) = \begin{cases} [\![l, \top]\!] & \text{if } l \not\preceq \boldsymbol{X}^{\boldsymbol{\mathcal{L}}} \\ [\![l, \boldsymbol{X}^{\boldsymbol{\mathcal{U}}}]\!] & \text{otherwise} \end{cases}$$

It is easy to see that this operator is well-defined and monotonic. First we show that $St([\![\bot, \top]\!]) = [\![l, \top]\!]$. Since $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ are cpos and the monotonicity of $A^{\boldsymbol{\mathcal{L}}}_{\top}$ and $A^{\boldsymbol{\mathcal{U}}}_{\bot}$ follows straightforwardly from the monotonicity of A, we can calculate $St^1([\![\bot, \top]\!])$ and $St^2([\![\bot, \top]\!])$ using the monotonic inductions of $A^{\boldsymbol{\mathcal{L}}}_{\top}$ and $A^{\boldsymbol{\mathcal{U}}}_{\bot}$, respectively.

Clearly, $A([\![\bot, \top]\!]) = [\![l, \top]\!]$ independently of which rule applies (although it must be the first). Therefore, $A^{\boldsymbol{\mathcal{L}}}_{\top}(\bot) = [\![l, \top]\!]^{\boldsymbol{\mathcal{L}}} = l$. Further, for $A([\![l, \top]\!])$, the second rule applies, thus $A([\![l, \top]\!]) = [\![l, [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}]\!] = [\![l, \top]\!]$. Hence, $A^{\boldsymbol{\mathcal{L}}}_{\top}(l) = [\![l, \top]\!]^{\boldsymbol{\mathcal{L}}} = l$, thus l is a fixpoint of $A^{\boldsymbol{\mathcal{L}}}_{\top}$. As a result, we obtain $St^1([\![\bot, \top]\!]) = l$.

Let us define $\bot_{\boldsymbol{\mathcal{U}}}$ to be the least element of the cpo $\boldsymbol{\mathcal{U}}$. Clearly, $\bot_{\boldsymbol{\mathcal{U}}} \in \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(S)$ by definition of the least element. Therefore, it must be the case that $l \not\preceq \bot_{\boldsymbol{\mathcal{U}}}$, otherwise the Interlattice Lb Property would hold for S. Consequently, $l \not\preceq [\![\bot, \bot_{\boldsymbol{\mathcal{U}}}]\!]^{\boldsymbol{\mathcal{L}}} \preceq \bot_{\boldsymbol{\mathcal{U}}}$. As a result, $A([\![\bot, \bot_{\boldsymbol{\mathcal{U}}}]\!]) = [\![l, \top]\!]$. Hence, $A^{\boldsymbol{\mathcal{U}}}_{\bot}(\bot_{\boldsymbol{\mathcal{U}}}) = [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}$. Further, if for $A([\![\bot, [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}]\!])$, the first rule applies, then $A([\![\bot, [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}]\!]) = [\![l, \top]\!]$. Otherwise, $A([\![\bot, [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}]\!]) = [\![l, [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}]\!] = [\![l, \top]\!]$. In summary, $A^{\boldsymbol{\mathcal{U}}}_{\bot}([\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}) = [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}$. Hence $St^2([\![\bot, \top]\!]) = [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}$. As a result, $St([\![\bot, \top]\!]) = [\![l, [\![l, \top]\!]^{\boldsymbol{\mathcal{U}}}]\!] = [\![l, \top]\!]$. Therefore,

$$St([\![\bot, \top]\!])^{\boldsymbol{\mathcal{L}}} = [\![l, \top]\!]^{\boldsymbol{\mathcal{L}}} = l$$

Define

$$U = Min([l, \top]_{\boldsymbol{\mathcal{U}}})$$

It must be the case that for every $s \in S$, there is an $u \in U$ such that $u \preceq s$. Otherwise, we would have that for every $t \in [l, s]_{\boldsymbol{\mathcal{U}}}$, there exists a $t' \in [l, s]_{\boldsymbol{\mathcal{U}}}$ such that $t' \preceq t$. Let us take a saturated chain C in $[l, s]_{\boldsymbol{\mathcal{U}}}$. Since $\boldsymbol{\mathcal{U}}$ is a co-cpo, C must have a greatest lower bound $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$. By our assumption that the Interlattice Lb Property holds for all chains, $l \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$. Moreover, clearly, $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C) \preceq s$, hence $\mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C) \in [l, s]_{\boldsymbol{\mathcal{U}}}$. Therefore, there must exist a $t' \in [l, s]_{\boldsymbol{\mathcal{U}}}$ such that $t' \preceq \mathrm{glb}_{\boldsymbol{\mathcal{U}}}(C)$. This contradicts the assumption that the chain was saturated.

Then it is easy to see that $\mathrm{lb}_{\boldsymbol{\mathcal{U}}}(U) \subseteq \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(S)$, since S is non-empty. Therefore, there is also no $u' \in \mathrm{lb}_{\boldsymbol{\mathcal{U}}}(U)$, such that $l \preceq u'$. Thus U must contain at least two elements.

Now we will show that for every $u \in U$, u must be a fixpoint of $A_l^{\mathcal{U}}$. In particular, it is easy to see that for $u \in U$, $A([\![l,u]\!]) = [\![l,u]\!]$. By condition 2 in the definition of a composition poset $l \preceq [\![l,u]\!]^{\mathcal{L}} \preceq [\![l,u]\!]^{\mathcal{U}} \preceq u$. Since u was minimal in $[l,\top]_{\mathcal{U}}$, this entails $[\![l,u]\!]^{\mathcal{U}} = u$. Therefore, $A_l^{\mathcal{U}}(u) = A^{\mathcal{U}}([\![l,u]\!]) = [\![l,u]\!]^{\mathcal{U}} = u$. Hence, u is a fixpoint of $A_l^{\mathcal{U}}$.

Since U contains exactly the minimal elements in $[l,\top]_{\mathcal{U}}$ and $A_l^{\mathcal{U}}$ is only defined on $[l,\top]_{\mathcal{U}}$, this implies that $A_l^{\mathcal{U}}$ has a least fixpoint iff U identifies a unique minimal element. However, we concluded before that U must contain at least two elements. Therefore $St^2([\![l,\top]\!])$ is not well-defined, and thus $St([\![l,\top]\!]) = St(St([\![\bot,\top]\!]))$ is not well-defined. □

Again, we notice that since Requirement 5 indirectly requires that $St(St([\![\bot,\top]\!]))$ is well-defined, it essentially implies the Interlattice Lb Property. In summary, every condition introduced in the generalized theory can be reduced to a consequence of the proposed requirements. Interestingly, we established in Section 5.5 that the generalized theory entails that the stable operator is monotonic, well-defined and internal on the cpo of A-prudent and A-reliable approximants. Now it straightforwardly follows that, as claimed, the alternative requirement for Requirement 5 would indeed result in an equivalent definition for the approximation space.

Bibliography

[1] Samson Abramsky and Achim Jung. Handbook of logic in computer science (vol. 3). chapter Domain Theory, pages 1–168. Oxford University Press, Inc., New York, NY, USA, 1994. ISBN 0-19-853762-X. URL http://dl.acm.org/citation.cfm?id=218742.218744.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Wilhelm Ackermann. Zum hilbertschen aufbau der reellen zahlen. *Mathematische Annalen*, 99:118–133, 1928. URL http://eudml.org/doc/159248.

[4] Peter Aczel. An introduction to inductive definitions. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 739–782. Elsevier, 1977.

[5] Peter Aczel. The type theoretic interpretation of constructive set theory: inductive definitions. In *Studies in Logic and the Foundations of Mathematics*, volume 114, pages 17–49. Elsevier, 1986.

[6] Rehan Abdul Aziz. Bound founded answer set programming. *CoRR*, abs/1405.3367, 2014. URL http://arxiv.org/abs/1405.3367.

[7] Nuel D. Belnap. A useful four-valued logic. In J. Michael Dunn and George Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37. Reidel, Dordrecht, 1977. Invited papers from the Fifth International Symposium on Multiple-Valued Logic, held at Indiana University, Bloomington, Indiana, May 13-16, 1975.

[8] Yi Bi, Jia-Huai You, and Zhiyong Feng. A generalization of approximation fixpoint theory and application. In Roman Kontchakov and Marie-Laure Mugnier, editors, *Web Reasoning and Rule Systems*, pages 45–59, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11113-1.

[9] Jasmin Christian Blanchette, Aymeric Bouzy, Andreas Lochbihler, Andrei Popescu, and Dmitriy Traytel. Friends with benefits: Implementing corecursion in foundational proof assistants. In *European Symposium on Programming*, pages 111–140. Springer, 2017.

[10] Bart Bogaerts. Weighted abstract dialectical frameworks through the lens of approximation fixpoint theory. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 2686–2693. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33012686. URL https://doi.org/10.1609/aaai.v33i01.33012686.

[11] Bart Bogaerts, Joost Vennekens, and Marc Denecker. On well-founded set-inductions and locally monotonic operators. *ACM Trans. Comput. Logic*, 17(4): 27:1–27:32, September 2016. ISSN 1529-3785. doi: 10.1145/2963096. URL http://doi.acm.org/10.1145/2963096.

[12] Bart Bogaerts, Angelos Charalambidis, Giannos Chatziagapis, Babis Kostopoulos, Samuele Pollaci, and Panos Rondogiannis. The stable model semantics for higher-order logic programming. *Theory and Practice of Logic Programming*, 24(4): 737–754, 2024. doi: 10.1017/S1471068424000231.

[13] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In Fangzhen Lin, Ulrike Sattler, and Mirosław Truszczyński, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, pages 102–111. AAAI Press, 2010. URL http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1294.

[14] Gerhard Brewka, Jörg Pührer, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Weighted abstract dialectical frameworks: Extended and revised report. *CoRR*, abs/1806.07717, 2018. URL http://arxiv.org/abs/1806.07717.

[15] Gerhard Brewka, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Weighted abstract dialectical frameworks. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16373.

[16] Luitzen Egbertus Jan Brouwer. *Over de grondslagen der wiskunde*. Maas & van Suchtelen, 1907.

[17] Wilfried Buchholz, Solomon Feferman, Wolfram Pohlers, and Wilfried Sieg. *Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoretical Studies*, volume 897 of *Lecture Notes in Mathematics*. Springer, 1981.

[18] Pedro Cabalar, David Pearce, and Agustín Valverde. Stable reasoning. *Journal of Applied Non-Classical Logics*, 27(3-4):238–254, 2017. doi: 10.1080/11663081.2018.1439358.

[19] Pedro Cabalar, Jorge Fandinno, Torsten Schaub, and Sebastian Schellhorn. Lower bound founded logic of here-and-there. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 509–525. Springer, 2019. doi: 10.1007/978-3-030-19570-0_34. URL https://doi.org/10.1007/978-3-030-19570-0_34.

[20] James Caldwell. Structural induction principles for functional programmers. *Electronic Proceedings in Theoretical Computer Science*, 136:16–26, December 2013. ISSN 2075-2180. doi: 10.4204/eptcs.136.2. URL http://dx.doi.org/10.4204/EPTCS.136.2.

[21] Angelos Charalambidis and Panos Rondogiannis. Categorical approximation fixpoint theory. In *European Conference on Logics in Artificial Intelligence*, pages 515–530. Springer, 2023.

[22] Angelos Charalambidis, Panos Rondogiannis, and Ioanna Symeonidou. Approximation fixpoint theory and the well-founded semantics of higher-order logic programs. *TPLP*, 18(3-4):421–437, 2018. doi: 10.1017/S1471068418000108. URL https://doi.org/10.1017/S1471068418000108.

[23] Patrick Cousot and Radhia Cousot. Constructive versions of tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43 – 57, 1979. doi: pjm/1102785059. URL https://doi.org/.

[24] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.

[25] Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, Gerda Janssens, and Marc Denecker. Predicate logic as a modeling language: the IDP system. In *Declarative Logic Programming: Theory, Systems, and Applications*, pages 279–323. 2018. doi: 10.1145/3191315.3191321. URL https://doi.org/10.1145/3191315.3191321.

[26] Marc Denecker. Extending classical logic with inductive definitions. In John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia

Palamidessi, Luís Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *CL*, volume 1861 of *LNCS*, pages 703–717. Springer, 2000. ISBN 3-540-67797-6.

[27] Marc Denecker. What's in a model? Epistemological analysis of logic programming. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *KR*, pages 106–113. AAAI Press, 2004. ISBN 1-57735-199-1.

[28] Marc Denecker and Eugenia Ternovska. Inductive situation calculus. *Artif. Intell.*, 171(5-6):332–360, 2007.

[29] Marc Denecker and Eugenia Ternovska. A logic of nonmonotonic inductive definitions. *ACM Trans. Comput. Log.*, 9(2):14:1–14:52, April 2008. ISSN 1529-3785. URL http://dx.doi.org/10.1145/1342991.1342998.

[30] Marc Denecker and Joost Vennekens. Well-founded semantics and the algebraic theory of non-monotonic inductive definitions. In Chitta Baral, Gerhard Brewka, and John S. Schlipf, editors, *LPNMR*, volume 4483 of *Lecture Notes in Computer Science*, pages 84–96. Springer, 2007. ISBN 978-3-540-72199-4. doi: 10.1007/978-3-540-72200-7_9. URL http://dx.doi.org/10.1007/978-3-540-72200-7_9.

[31] Marc Denecker and Joost Vennekens. The well-founded semantics is the principle of inductive definition, revisited. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *KR*, pages 1–10. AAAI Press, 2014. ISBN 978-1-57735-657-8. URL http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7957.

[32] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, volume 597 of *The Springer International Series in Engineering and Computer Science*, pages 127–144. Springer US, 2000. ISBN 978-1-4613-5618-9. doi: 10.1007/978-1-4615-1567-8_6. URL http://dx.doi.org/10.1007/978-1-4615-1567-8_6.

[33] Marc Denecker, Maurice Bruynooghe, and Victor Marek. Logic programming revisited: Logic programs as inductive definitions. *ACM Trans. Comput. Log.*, 2 (4):623–654, 2001.

[34] Marc Denecker, Nikolay Pelov, and Maurice Bruynooghe. Ultimate well-founded and stable semantics for logic programs with aggregates. In Philippe Codognet, editor, *Logic Programming, 17th International Conference, ICLP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings*, volume 2237 of *Lecture Notes in Computer Science*, pages 212–226. Springer, 2001. ISBN 3-540-42935-2. doi: 10.1007/3-540-45635-X. URL https://doi.org/10.1007/3-540-45635-X.

[35] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Uniform semantic treatment of default and autoepistemic logics. *Artif. Intell.*, 143(1):79–122, 2003. URL http://dx.doi.org/10.1016/S0004-3702(02)00293-X.

[36] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation*, 192(1):84–121, July 2004. doi: 10.1016/j.ic.2004.02.004. URL https://lirias.kuleuven.be/handle/123456789/124562.

[37] Marc Denecker, Maurice Bruynooghe, and Joost Vennekens. Approximation fixpoint theory and the semantics of logic and answer set programs. In *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, pages 178–194. Springer, 2012.

[38] Kees Doets and Jan Eijck. *The Haskell road to logic, maths and programming*. College Publications, 2004.

[39] Paul Downen and Zena M. Ariola. Classical (co)recursion: Programming. *CoRR*, abs/2103.06913, 2021. URL https://arxiv.org/abs/2103.06913.

[40] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[41] Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artif. Intell.*, 175(1):278–298, 2011. doi: 10.1016/j.artint.2010.04.002. URL http://dx.doi.org/10.1016/j.artint.2010.04.002.

[42] Solomon Feferman. Formal theories for transfinite iterations of generalised inductive definitions and some subsystems of analysis. In A. Kino, J. Myhill, and R.E. Vesley, editors, *Intuitionism and Proof theory*, pages 303–326. North Holland, 1970.

[43] Paolo Ferraris. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic (TOCL)*, 12(4):1–40, 2011.

[44] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A new perspective on stable models. In Manuela M. Veloso, editor, *IJCAI*, pages 372–379, 2007.

[45] Melvin Fitting. A Kripke-Kleene semantics for logic programs. *The Journal of Logic Programming*, 2(4):295 – 312, 1985. ISSN 0743-1066. doi: http://dx.doi.org/10.1016/S0743-1066(85)80005-4. URL http://www.sciencedirect.com/science/article/pii/S0743106685800054.

[46] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *ICLP/SLP*, pages 1070–1080. MIT Press, 1988. ISBN 0-262-61056-6. URL http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.6050.

[47] Michael Gelfond and Yuanlin Zhang. Vicious circle principle, aggregates, and formation of sets in asp based languages. *Artificial Intelligence*, 275:28–77, 2019.

[48] K. Gödel. Zum intuitionistischen aussagenkalkül. *Anzeiger der Akademie der Wissenschaften in Wien*, 69:65–66, 1932.

[49] Egbert Harzheim. *Ordered sets*, volume 7. Springer Science & Business Media, 2005.

[50] Jesse Heyninck, Ofer Arieli, and Bart Bogaerts. Non-deterministic approximation fixpoint theory and its application in disjunctive logic programming. *Artificial Intelligence*, 331:104110, 2024. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2024.104110. URL https://www.sciencedirect.com/science/article/pii/S0004370224000468.

[51] Paul Hudak. Conception, evolution, and application of functional programming languages. *ACM Comput. Surv.*, 21(3):359–411, 1989. ISSN 0360-0300. doi: 10.1145/72551.72554. URL http://dx.doi.org/10.1145/72551.72554.

[52] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

[53] David B. Kemp and Peter J. Stuckey. Semantics of logic programs with aggregates. In Vijay A. Saraswat and Kazunori Ueda, editors, *ISLP*, pages 387–401. MIT Press, 1991. ISBN 0-262-69147-7.

[54] Stephen Cole Kleene. On the forms of the predicates in the theory of constructive ordinals. (66):41–58, 1944.

[55] Stephen Cole Kleene. *Introduction to Metamathematics*. Van Nostrand, 1952.

[56] Dexter Kozen and Alexandra Silva. Practical coinduction. *Mathematical Structures in Computer Science*, 27(7):1132–1152, 2017.

[57] Lengning Liu, Enrico Pontelli, Tran Cao Son, and Miroslaw Truszczynski. Logic programs with abstract constraint atoms: The role of computations. *Artif. Intell.*, 174(3-4):295–315, 2010. doi: 10.1016/j.artint.2009.11.016. URL https://doi.org/10.1016/j.artint.2009.11.016.

[58] Victor W Marek and Jeffrey B Remmel. Set constraints in logic programming. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 167–179. Springer, 2004.

[59] Đorđe Marković, Simon Vandevelde, Linde Vanbesien, Joost Vennekens, and Marc Denecker. An epistemic logic for modeling decisions in the context of incomplete knowledge. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, page 789–793, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400702433. doi: 10.1145/3605098.3636176. URL https://doi.org/10.1145/3605098.3636176.

[60] George Markowsky. Chain-complete posets and directed sets with applications. *Algebra Universalis*, 6(1):53–68, December 1976. ISSN 1420-8911. doi: 10.1007/bf02485815. URL https://doi.org/10.1007/bf02485815.

[61] Per Martin-Löf. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In J.e. Fenstad, editor, *Second Scandinavian Logic Symposium*, pages 179–216, 1971.

[62] Robert C. Moore. A Formal Theory of Knowledge and Action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Springer-Verlag, 1985.

[63] Yiannis N. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland Publishing Company, Amsterdam- New York, 1974.

[64] Christine Paulin-Mohring. Inductive definitions in the system coq rules and properties. In *International Conference on Typed Lambda Calculi and Applications*, pages 328–345. Springer, 1993.

[65] Lawrence C Paulson. A fixedpoint approach to (co) inductive and (co) datatype definitions. In *Proof, Language, and Interaction*, pages 187–212, 2000.

[66] Lawrence C. Paulson and Andrew W. Smith. Logic programming, functional programming, and inductive definitions. *CoRR*, cs.LO/9301109, 1993. URL https://arxiv.org/abs/cs/9301109.

[67] Nikolay Pelov. *Semantics of Logic Programs with Aggregates*. PhD thesis, K.U.Leuven, Leuven, Belgium, April 2004.

[68] Nikolay Pelov, Emmanuel De Mot, and Maurice Bruynooghe. A comparison of logic programming approaches for representation and solving of constraint satisfaction problems. In *Proceedings of the 8th International Workshop on Nonmonotonic Reasoning NMR'2000*, pages 1–10, 2000.

[69] Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *TPLP*, 7(3):301–353, 2007. doi: 10.1017/S1471068406002973. URL http://dx.doi.org/10.1017/S1471068406002973.

[70] Rozsa Peter. Konstruktion nichtrekursiver funktionen. *Mathematische Annalen*, 111:42–60, 1935.

[71] Samuele Pollaci. A category-theoretic perspective on approximation fixpoint theory. *arXiv preprint arXiv:2502.09234*, 2025.

[72] Samuele Pollaci, Babis Kostopoulos, Marc Denecker, and Bart Bogaerts. A category-theoretic perspective on higher-order approximation fixpoint theory. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 317–330. Springer, 2024.

[73] Eric S. Roberts. *Thinking recursively*. Wiley, 1986. ISBN 978-0-471-81652-2.

[74] Raphael M. Robinson. Recursion and double recursion. *Bulletin of the American Mathematical Society*, 54:987–993, 1948.

[75] Manuel Rubio-Sanchez. *Introduction to recursive programming*. CRC Press, 2017.

[76] Vlad Rusu and David Nowak. Defining corecursive functions in Coq using approximations. In Karim Ali and Jan Vitek, editors, *36th European Conference on Object-Oriented Programming, ECOOP 2022, June 6-10, 2022, Berlin, Germany*, volume 222 of *LIPIcs*, pages 12:1–12:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPIcs.ECOOP.2022.12. URL https://doi.org/10.4230/LIPIcs.ECOOP.2022.12.

[77] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4), 2009. doi: 10.1145/1516507.1516510. URL http://doi.acm.org/10.1145/1516507.1516510.

[78] Dana Scott. Data types as lattices. In *Proceedings of the International Summer Institute and Logic Colloquium*, volume 499 of *Lecture Notes in Mathematics*, page 579–651. Springer-Verlag, 1975.

[79] Luke Simon, Ajay Mallya, Ajay Bansal, and Gopal Gupta. Coinductive logic programming. In *Logic Programming: 22nd International Conference, ICLP 2006, Seattle, WA, USA, August 17-20, 2006. Proceedings 22*, pages 330–345. Springer, 2006.

[80] Robert I. Soare. *Recursively enumerable sets and degrees - a study of computable functions and computability generated sets*. Perspectives in mathematical logic. Springer, 1987. ISBN 978-3-540-15299-6.

[81] Tran Cao Son, Enrico Pontelli, and Phan Huy Tu. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research*, 29:353–389, 2007.

[82] Clifford Spector. Inductively defined sets of natural numbers. In *Infinitistic Methods (Proc. 1959 Symposium on Foundation of Mathematis in Warsaw)*, pages 97–102. Pergamon Press, Oxford, 1961.

[83] Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artif. Intell.*, 205:39–70, 2013. doi: 10.1016/j.artint.2013.09.004. URL http://dx.doi.org/10.1016/j.artint.2013.09.004.

[84] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 1955.

[85] Linde Vanbesien, Maurice Bruynooghe, and Marc Denecker. Analyzing semantics of aggregate answer set programming using approximation fixpoint theory. *Theory and Practice of Logic Programming*, 22(4):523–537, 2022.

[86] Linde Vanbesien, Bart Bogaerts, and Marc Denecker. Approximation fixpoint theory with refined approximation spaces, 2024. URL https://arxiv.org/abs/2506.16294.

[87] Linde Vanbesien, Samuele Pollaci, Bart Bogaerts, and Marc Denecker. Towards a unifying view on monotonic constructive definitions. In Reneta P. Barneva, Valentin E. Brimkov, Claudio Gentile, and Aldo Pacchiano, editors, *Artificial Intelligence and Image Analysis*, pages 218–235, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-63735-3.

Curriculum Vitae

Linde Vanbesien was born in Roeselare, Belgium, on September 22nd 1997. In 2018 and 2020, she respectively obtained her B.Sc. and M.Sc. degree in engineering science in the field of Computer Science at KU Leuven. Afterwards, she started working towards a PhD in Engineering Science at KRR (Knowledge Representation and Reasoning), a sub-group of the DTAI-research group (Declarative Languages and Artificial Intelligence) at the Computer Science Department at KU Leuven. In 2023 this transitioned into a joint doctoral program with VUB, where she works towards a PhD in Science at the AILab (Artificial Intelligence Lab). Her research focuses on the formalization of different types of constructive knowledge, including, but not limited to, Approximation Fixpoint Theory and its applications, limits and extensions.

List of publications

Peer-Reviewed Papers Included in this Thesis

- Linde Vanbesien, Samuele Pollaci, Bart Bogaerts, and Marc Denecker. Towards a unifying view on monotonic constructive definitions. In Reneta P. Barneva, Valentin E. Brimkov, Claudio Gentile, and Aldo Pacchiano, editors, *Artificial Intelligence and Image Analysis*, pages 218–235, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-63735-3
- Linde Vanbesien, Maurice Bruynooghe, and Marc Denecker. Analyzing semantics of aggregate answer set programming using approximation fixpoint theory. *Theory and Practice of Logic Programming*, 22(4):523–537, 2022

Unpublished Papers Included in this Thesis

- Linde Vanbesien, Bart Bogaerts, and Marc Denecker. Approximation fixpoint theory with refined approximation spaces, 2024. URL `https://arxiv.org/abs/2506.16294`

Peer-Reviewed Co-authored Papers Not Included in this Thesis

- Đorđe Marković, Simon Vandevelde, Linde Vanbesien, Joost Vennekens, and Marc Denecker. An epistemic logic for modeling decisions in the context of incomplete knowledge. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, page 789–793, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400702433. doi: 10.1145/3605098.3636176. URL `https://doi.org/10.1145/3605098.3636176`



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DTAI
Celestijnenlaan 200A box 2402
B-3001 Leuven
linde.vanbesien@cs.kuleuven.be