New Language Constructs and Inferences for the Knowledge Base Paradigm

A business and multi-agent perspective

Pieter Van Hertum

Supervisors:
Prof. dr. M. Denecker
Prof. dr. ir. G. Janssens

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

October 2016

New Language Constructs and Inferences for the Knowledge Base Paradigm

A business and multi-agent perspective

Pieter VAN HERTUM

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

October 2016

Preface

Four years (approximated, which I can do now, since I will get an engineering degree), that I have spent in this research group, working on these interesting subjects, meeting a lot of interesting people, will now come to an end with writing this preface. It were amazing years, filled with joys and frustrations that were new to a recently graduated master's student. I learned the frustration of searching for weeks on end on a complicated problem and I learned the joy of waking up in the middle of the night with an idea that would solve it all. I learned the joy of compressing all the work I had spent months on into one paper, and the frustration of getting it rejected for parts of the problem I did not think about. I learned the frustration of having to rewrite the same piece of text again and again and again, but then the fantastic feeling if it finally becomes this really nice compact clear flow of ideas and it gets accepted for publication. These four interesting years and the resulting thesis would have never been possible if it was not for a whole bunch of people.

I would have never even started working in this subject if it wasn't for my supervisor **Marc**. Marc, your incredible amount of enthusiasm for your field of research is jaw-dropping and very inspiring[1]. When doing my master's thesis in your group, you sparked me with (a bit of) the same enthusiasm which made me want to pursue this PhD in Knowledge Representation, and which gave me the motivation to keep wanting it when things were not going as planned. Thank you for the many long discussions and ideas that have led to the research in this thesis. And finally, thank you for teaching me a whole new level of

[1]I know you don't like me using very in a text, but this time it was very much needed.

deadline-driven research, where almost half of a paper can be researched and written in the night before the deadline.

Halfway through my PhD, Marc got support in guiding me by **Gerda**. Thank you Gerda, for guiding me in the more application-oriented topics. Thank you for taking the time for proof-reading all my texts, even if the comments of last time sometimes weren't incorporated as well as you would have liked. And foremost, thank you for taking time to assist Marc in the management of my work, to stay grounded and to make sure things were planned correctly and executed according to plan (something we both were not so great at).

While many say that doing a PhD is a lonesome job, those people have not done their PhD in the KRR research group at KU Leuven. Being able to do your research in a group of (mostly) likeminded people makes it much more rewarding and motivating. Thank you **Broes, Stef, Bart, Joachim, Jo, Ingmar, Matthias, Ruben, Laurent** and **Tim** for the interesting (albeit sometimes extremely long) discussions, fun trips to Helsinki, Kasterlee, Istanbul, Westmalle, New York,. . . , and our very fun board game nights.

During my PhD I got the chance to travel to beautiful places to meet interesting people. There are a few people that I would like to thank in particular with helping in the forming of this PhD. Thank you to the members of my **jury** for traveling to Leuven (twice!) and/or for taking all the time to read this work and give detailed interesting feedback on it. **Vaishak**, thanks for guiding me into the complex field of modal logics and thank you for all ideas. **Marcos**, thank you for not agreeing to my ideas, since many of the results in this work come from those discussions and thank you for all the productive and fun trips to Luxembourg.

Outside of work, there are many friends I would like to thank here for relieving me of the grasp that thinking about "if someone knows is someone only knows whether something is known by everyone does he then know if this can be common knowledge" can get on your brain. Thank you **Ruben, Filip, Mariën, Derde, Wouter, Cois, Sanne, Vincent, . . .** [2] and the people in **Poseidon** for the fun times outside of work: playing games, drinking beers, taking trips, going to festivals, diving, ... A special thank you goes to **Cie** for my Master's degree.

To my **parents**: heel erg bedankt voor al de steun alle 9 jaar (en de 18 daarvoor) dat ik gespendeerd heb werkend naar dit doctoraat. Bedankt voor alle steun, zowel moreel als financieel. Ik kan met heel grote zekerheid zeggen dat ik hier nooit gestaan had zonder jullie. Dankjewel!

[2]Apologies to the people I might have forgotten, I mean nothing by it :)

To conclude, there is one more person to thank. As stated above, my reasoning skills are better than my planning skills, but luckily I had someone who always stood by me and helped through this. **Lieze**, thank you for helping me plan, and for being there when my planning failed and I had to do the deadline-driven research our research group was famous for. Thank you for always being there, listening to me pitching ideas (even if you had no clue what they were about), listening to me complaining and motivating me to push through when times were though and I couldn't see the forest for the trees. Ik zie je graag!

Abstract

In Artificial Intelligence, the scientific field of Knowledge Representation and Reasoning (KRR) is concerned with developing formal languages, to represent knowledge, and inference methods to solve tasks using that knowledge. Most of the existing approaches develop a formal language (a logic) together with an inference, to solve some type of computational task. The recently proposed Knowledge Base (KB) paradigm applies a strict separation of concerns to information and problem solving, based on the idea that knowledge is completely independent of the computational task it is used for. A KB system allows information to be stored in a knowledge base, and provides a range of inference methods, all using the same knowledge base. With these inference methods, various types of problems and tasks can be solved using the same knowledge base.

In this text we study this paradigm in detail and we prove the hypothesis that this approach has many advantages over standard declarative paradigms. We use an implementation of a KB system: the IDP system with corresponding language FO($\cdot$) to model challenging and interesting applications. These applications show the need to provide extensions: we develop new language constructs and formalize new inferences.

In the first part of this work, we study two applications from industry. We first look at interactive product configuration. In these interactive configuration problems, a configuration of interrelated objects under constraints is searched, where the system assists the user in reaching an intended configuration. We show that multiple functionalities in this domain can be achieved by applying different forms of inferences on a formal specification of the configuration domain. To this goal, a set of new, derived inferences are defined. A second application is the car-rental system, a prototypical example from the domain of Business Rules: rule-based systems that are used in industry for knowledge intensive applications. We investigate whether all necessary knowledge can be represented in our knowledge Representation language FO(·). We propose an extension of the concept of inductive definitions in the form of a "*new*"-operator to model knowledge about situations where a new object is created, for which a good formalism was not available in our framework.

In the second part, we focus on applying the ideas of the KB paradigm and the KB system IDP in a security context, by modelling Access Control applications. We use a KB system in a distributed way to verify if a certain agent has access, based on the policies of other agents. We develop a generalisation of autoepistemic logic to a distributed setting as a language for these distributed policies. Distributed autoepistemic logic (dAEL) is equipped with tools to express references to the knowledge of other agents. We study generalisations of well-known semantics of autoepistemic logic, using Approximation Fixpoint Theory (AFT). dAEL assumes complete mutual introspection between agents, every agent knows exactly what every other agent knows. This is a reasonable assumption when modelling public statements that an agent makes, but it is not reasonable for every multi-agent application.

In the last part of this thesis we present another new logic: $\mathcal{COL}_m$, an extension of first order logic with constructs for a knowledge modality K_A for each agent, a common knowledge modality C and an only knowing modality O_A for each agent. We study how these modalities behave together and develop a new semantic structure that can resolve formulas of this language, without assuming any introspection.

Samenvatting

Het doel van *kennisrepresatie en redeneren* (KRR), een onderzoeksveld in Artificiële Intelligentie is het ontwikkelen van formele talen om kennis mee voor te stellen op een wiskundige manier, en het ontwikkelen van inferentie methoden zodat verschillende taken kunnen worden opgelost met behulp van deze kennis.

Bestaande aanpakken in KRR ontwikkelen een formele taal (een logica) met een bijhorende inferentie, specifiek toegepast op een bepaald soort redeneertaak. Het Kennisbank (KB) paradigma pakt dit anders aan. Er is een strikte scheiding tussen de relevante kennis en waar het voor gebruikt wordt. De achterliggende visie is dat domeinkennis compleet onafhankelijk is van de redeneertaak waarvoor het gebruikt dient te worden. In een KB systeem wordt informatie opgeslagen in een centraal beheerde kennisbank, waarbij een verzameling inferenties worden aangeboden die redeneertaken kunnen oplossen, gebruikmakend van deze kennisbank.

In deze tekst bestuderen we dit paradigma en we onderzoeken en bewijzen dat deze aanpak een heel aantal voordelen biedt ten opzichte van eerdere aanpakken. We maken gebruik van een bestaand KB systeem: het IDP systeem, met eigen KR taal FO($\cdot$) (eerste-orde logica met een verzameling uitbreidingen). We gebruiken dit systeem om verschillende interessante en complexe toepassingen te modelleren. Vanuit deze toepassingen zien we de nood tot het uitwerken verschillende uitbreidingen. We definiëren nieuwe taalconcepten en ontwikkelen nieuwe inferentietechnieken.

In het eerste deel van deze thesis bekijken we twee toepassingen uit de industrie.

De eerste gaat over interactieve configuratie van producten en systemen. In interactieve configuratie problemen wordt er een configuratie gezocht van verschillende gerelateerde objecten die aan een verzameling voorwaarden moeten voldoen. Het systeem begeleidt de gebruiker in deze configuratietaak en werkt samen met de gebruiker om een gewenste configuratie te bereiken. We bekijken deze toepassingen en tonen aan dat verschillende functionaliteiten bereikt kunnen worden door inferenties toe te passen op een formele specificatie van de domeinkennis over de configuratie. We definiëren een verzameling nieuwe, afgeleide inferenties die deze nieuwe functionaliteiten voorzien. De tweede toepassing die we bekijken is een systeem dat een autoverhuur bedrijf ondersteund. Dit is een voorbeeld dat prototypisch is in de wereld van de Business Rules: regelgebaseerde systemen die vaak in de industrie gebruikt worden voor kennisintensieve toepassingen. We onderzoeken of alle relevante domeinkennis in onze taal FO$(\cdot)$ kan worden uitgedrukt. We definiëren een uitbreiding op het concept van inductieve definities in onze taal, die het toelaat om nieuwe objecten te creeëren en introduceren een "*new*"-operator, aangezien dit formalisme ontbrak in de huidige taal.

Het tweede deel van deze thesis bekijken we Access Control (of toegangsbeheer) toepassingen vanuit het oogpunt van het KB paradigma. We gebruiken het IDP systeem en kijken naar uitbreidingen voor de taal om deze bruikbaar te maken in een security context, voor deze Access Control toepassingen.

We onderzoeken hoe we een KB systeem op een gedistribueerde manier (onder verschillende agenten) kunnen gebruiken om uit te zoeken welke agent toegang krijgt tot welke bestanden, gebaseerd op het beleid van de andere agenten. We definiëren dAEL: distributed autoepistemic logic, een veralgemening van autoepistemische logica om dergelijk gedistribueerd beleid in voor te stellen. Distributed autoepistemic logic (dAEL) is uitgerust met taalconstructies om te refereren naar kennis van andere agenten. We bekijken semantieken voor deze logica, gebaseerde op gekende semantieken voor autoepistemische logica. Hiervoor gebruiken we het algebraïsch framework Approximation Fixpoint Theory (AFT). dAEL veronderstelt dat de verschillende agenten open en eerlijk comminuceren en bijgevolg perfect weten wat elke andere agent weet en niet weet. Deze veronderstelling is redelijk voor de toepassingen die we onderzocht hebben, maar is dit niet in het algemeen.

In het laatste deel van deze thesis stellen we $\mathcal{COL}_m$ voor: een logica gebaseerd op eerste-orde logica met taalconstructies voor verschillende modaliteiten: kennis van een agent (K_A), common knowledge (C) en beperkingen van kennis van een agent (Only knowing: O_A). We onderzoeken deze modaliteiten, bekijken hoe ze zich onderling verhouden en ontwikkelen een nieuwe semantische structuur die toelaat formules van deze vorm te interpreteren. In deze logica hebben we geen vooropgestelde veronderstellingen zoals introspectie.

Abbreviations

AEL	Autoepistemic Logic. xvi, 23, 27, 80
AFT	Approximation Fixpoint Theory. xvi, 7, 25–27
dAEL	Distributed Autoepistemic Logic. 80, 82
DBP	Distributed Belief Pair. 83, 84, 86, 87, 95
DPWS	Distributed Possible World Structure. 83, 84, 92, 94
FO	First-order Logic. 7–9, 11, 14–16, 21, 25, 72, 81
IC	Interactive Configuration. 31
KB	Knowledge Base. xvi, 20–22
KBS	Knowledge Base System. 20, 57
KR	Knowledge Representation. 21
KRR	Knowledge Representation and Reasoning. 1, 2
LTC	Linear Time Calculus. 18
PWS	Possible World Structure. 23, 24, 28
SGN	Strong Global Negative. 98

List of Symbols

$comp(P, \Delta)$	The completion of P as defined by Δ, page 18
$\mathcal{A}$	A set of agents in a modal logic, page 81
$\bigwedge S$	The greatest lower bound of S, page 25
$\bigvee S$	The least upper bound of S, page 25
$\bot$	The least element in a lattice, page 25
$Def(\Delta)$	The defined symbols in a (inductive) defintion, page 17
Δ	A (inductive) definition, page 17
δ	A rule in a (inductive) defintion, page 17
$\mathrm{FO}(\cdot)$	A set of extensions for first-order logic, page 14
$\mathrm{FO(ID)}$	First-order logic, extended with (inductive) definitions, page 17
$\mathcal{L}_{FO}$	The language of first-order logic, page 11
$\geq_p$	The precision order between (partial) structures, page 12
$\geq_p$	The precision relation between worlds, page 110
$\geq_t$	The truth order between (partial) structures, page 12

κ	A successor ordinal, page 108
λ	A limit ordinal, page 108
λ-$\mathcal{COL}_m^2$	The logic consisting of the language $\mathcal{COL}_m$ and a two-valued valuation using λ-worlds, page 126
$\langle v_\alpha \rangle_{\alpha<\mu}$	A sequence of length μ where element α is v_α, page 108
$\leq_K$	The knowledge order between possible world structures, page 24
Σ_{LTC}	A time-dependent LTC vocabulary, page 18
$\mathbb{W}$	The set of all possible objective worlds as in [Belle and Lakemeyer, 2015], page 131
$\models$	The satisfaction relation, $S \models T$ means that S is a model of T, page 12
μ	An arbitrary ordinal, page 108
μ-$\mathcal{CL}_m$	The logic consisting of the language $\mathcal{CL}_m$ and a valuation using μ-worlds, page 117
μ-$\mathcal{COL}_m$	The logic consisting of the language $\mathcal{COL}_m$ and a valuation using μ-worlds, page 126
ω	The lowest transfinite ordinal number and the order type of the natural numbers under their usual linear ordering, page 119
$Par(\Delta)$	The open symbols in a (inductive) defintion, page 17
$\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$	The language of multi-agent modal logic with common knowledge, only knowing and public announcements, page 143
$\mathcal{L}_{Prop}$	The language of propositional logic, page 11
$\mathcal{S}$	A partial structure, page 12
$\mathcal{S}_D$	The domain structure of a (partial) structure $\mathcal{S}$, page 14
Q	A possible world structure, page 23
S	A structure, page 9

$(w)_\alpha$	Given a sequence w, $(w)_\alpha$ is element α of that sequence, page 109
S_{LTC}	A time-dependent LTC structure, page 18
τ	The translation functinon from dAEL to AEL, where a subscript indicates the type of the output of the translation function, page 90
$\mathbb{T}$	The set of terms, page 10
$\top$	The greatest element in a lattice, page 25
$\mathcal{B}$	A Distributed Belief Pair, page 84
$\mathcal{B}^c$	The conservative bounds of a DBP $\mathcal{B}$ (a DPWS), page 84
$\mathcal{B}^l$	The liberal bounds of a DBP $\mathcal{B}$ (a DPWS), page 84
$\mathcal{D}^*_\mathcal{T}$	The approximator for a distributed theory $\mathcal{T}$, page 86
$\mathcal{Q}$	A Distributed Possible World Structure, page 83
$\mathcal{D}_\mathcal{T}$	The knowledge revision operator for a distributed theory $\mathcal{T}$, page 85
$\mathcal{D}^{st}_\mathcal{T}$	The stable operator for a distributed theory $\mathcal{T}$, page 86
Σ	A vocabulary, with subscripts P, F and T to denote the set of predicates, functions and types (in a typed logic), page 8
$\mathcal{W}^{in}_\lambda$	The set of informed λ-worlds, page 142
$\mathcal{W}^{ni}_\lambda$	The set of negative introspective λ-worlds, page 142
$\mathcal{W}^{pi}_\lambda$	The set of positive introspective λ-worlds, page 142
$\mathcal{W}_\mu$	The set of μ-worlds, page 109
A^w	The epistemic state of an agent A in a μ-world w, page 109
Agg	An aggregate function. $Agg \in \{Min, Max, Sum, Card, Prod\}$, page 16
D_S	The domain in a structure S, page 9
D_T	The revision operator for a AEL theory T, page 25
$glb(S)$	The greatest lower bound of S, page 25

I	A set of integers $\{1, \ldots, n\}$ representing the agents in [Belle and Lakemeyer, 2015], page 131
L	A lattice, page 25
L^2	The bilattice of L, page 25
L^c	The consistent elements of L^2, page 25
$lub(S)$	The least upper bound of S, page 25
$MD(\varphi)$	The modal depth of a formula φ, page 119
$R(w)$	The reduction function, the union of all R_μ for all μ, page 110
$R_\mu(w)$	The reduction of $\mu + 1$ world w, page 110
S^A	The stable operator for an approximator A, page 26
T	A theory, page 11
$T_\mathcal{S}$	The associated theory of a (partial) structure, page 14
w^{obj}	The objective state of a μ-world w, page 109
X_α	The α-extraction from a μ-world, page 111
$\mathcal{L}_k$	The language of autoepistemic logic, page 23
$\mathcal{CL}_m$	The language of multi-agent modal logic with common knowledge, page 105
$\mathcal{CO}^\gamma\mathcal{L}_m$	The language of multi-agent modal logic with common knowledge and limited only-knowing up until depth γ, page 136

Contents

List of Figures

1 Introduction

Key to creating artificial intelligence is making machines "understand" information. Solving problems often requires an extensive amount of knowledge about a certain field of discourse. The field of Knowledge Representation and Reasoning (KRR) is concerned with representing information about the world in a form such that it can be comprehended by a machine and studies how this knowledge can be used for reasoning.

The first aspect of KRR is about developing languages to represent knowledge. To do this, a language has to be a formal, mathematical, language (a *logic*), yet it has to be natural and readable since humans have to express knowledge in this language. Many logics have been studied for this task, some of which will be discussed in this thesis. A well-known logic, prominently featured in this thesis is *first-order logic*. First-order logic is a mathematical language with operators (such as *and*, *or*, *not*, ...) and quantifications (such as *for all* and *there exists*) that allow to combine (atomic) facts to form more complex statements. In first-order logic, statements such as "All men are mortal" and "Socrates is a man" can be expressed as:

$$\forall x : Man(x) \Rightarrow Mortal(x).$$

$$Man(Socrates).$$

The second aspect is about studying how specifications in such formal languages can be used to solve complex problems or how they can be used for reasoning.

Inferences or reasoning tasks take a specification in a formal language as input and use this for reasoning. For example, the *deduction* inference can be used to derive from the above specification that Socrates is mortal ($Mortal(Socrates)$). The deduction inference is only one of many useful possible inference tasks that are studied in the field. In this thesis we will discuss many more, such as for example *modelgeneration* (generate a situation or an example, i.e. a model, that is possible given the specification formulated in the language).

A third aspect is developing reasoning machines or algorithms implementing these inferences. Combining these three aspects allows to build systems where a user can specify the relevant knowledge in a readable formal language, and the system can use this knowledge to solve real-life problem without the need of the user to state how the system should solve these problems. Many KRR systems have been built, where the three aspects are tightly connected: a system is developed supporting a specific inference with a language developed to state knowledge, that is to be used with that specific inference in mind.

The vision behind the *Knowledge Base Paradigm* is that while all three aspects are crucial, they should not be intertwined in any way. Information is independent of the inference it is used for and has nothing to do with the implementation wherein it is used. The goal is to develop a *Knowledge Base System (KBS)*, where a user (often an expert in the relevant domain) stores his knowledge about the domain of discourse and the system provides the user with a whole set of inferences that he can use to solve different tasks in that domain. A good example of an application of a KBS is a course scheduling task at a school. The domain knowledge consists of constraints on the availability of teachers, rooms, . . . , and of information on the groups of students, courses that have to be taught, etc. This information can be used to make a schedule at the start of the year, but is also relevant when a teacher falls ill and a revision of the schedule has to be made.

In this thesis we focus on the language FO($\cdot$) and the IDP Knowledge Base System that implements a number of inferences for FO($\cdot$). Important when developing such a system is to have a language that allows to represent as many applications as possible in a precise, concise but natural way and to support many reasoning tasks by logical inferences. Since the only way to investigate the maturity of such a system (“Is the language rich enough to express real-life knowledge?”, “Are there enough inferences available to be used in real-life applications?”,. . .) is to take it out into the world and try it, which is what we have done in this thesis. We investigate the applicability of the IDP system in different contexts and study what language constructs and reasoning tasks are still lacking. We formulate extensions to the knowledge representation language FO($\cdot$) and formulate new inferences that allow to apply the IDP system in these application domains.

Studying applications from a KB point of view is two-fold: it is not only a test for the idea of the Knowledge Base Paradigm, it also studies how information is used in these applications, what kind of language constructs are relevant and study how they work. The applications motivating this thesis can be divided in two subcategories. First, we have business applications. In the first two chapters, we study interactive configuration as an application of the KB paradigm and look at an application from the domain of Business Rules: a formalism that is often used in industry for knowledge intensive applications. Afterwards, in the last two chapters, we look at modal logics. Motivated by an application in access control we develop modal logics for representing policies of agents in access control applications. We also propose general modal logics, that do not make any assumptions and have a rich language to express many modalities, for a wide range of applications.

The main contributions of the presented research are as follows:

- We present our case study of applying the KBS to the field of *interactive configuration*. We study the feasibility and usefulness of using a KBS in this setting. We identify eight subtasks an interactive configuration system should support, such as “Acquiring information from the user”, “Checking the consistency of a given answer by the user” and “Autocompletion of a partial configuration” and formalize them all as logical inferences on a centrally maintained knowledge base. We developed a prototype of an implementation, using the domain knowledge from a configuration application provided to us by a banking company and evaluate our approach using the criteria from knowledge-based configuration research and compare to other approaches using these same criteria.

  Summarized: **Studying how to apply a KBS to the interactive configuration domain, we formalize eight inferences on a central knowledge base implementing all relevant subtasks of an interactive configuration system.**

  This work was published as a conference paper at the Practical Aspects of Declarative Languages conference in [Van Hertum et al., 2016b] and later an extended version was published as a journal paper in Theory and Practice of Logic Programming [Van Hertum et al., 2016c].

- The domain of Business Rules is well-represented in industry for handling knowledge intensive applications. They often use a rule-based format to encode information. In order to compare our knowledge based approach with this field, we study a prototypical example from the Business Rules domain: the EU-Rent Car Rental company. EU-Rent is a fictitious car rental company spread over different countries, with a collection of cars spread over multiple branches. We study how the information could be formalized and implemented two relevant use cases. We plan reservations using a specification representing the rules and constraints relevant for a car rental company and we study how to represent the knowledge of purchasing a new car and adding it to the system. We propose a new inference for the first use case and a set of language extensions for the second. The research that is presented in Chapter 3 was a first step in exploring the problem domain and the motivation for further work, done by Bogaerts et al. in [Bogaerts et al., 2014c] and [Bogaerts et al., 2014b].

Summarized: **To represent rule-based knowledge in the IDP KBS, we introduce a new language construct allowing to create new domain elements and tested this approach in a use case of the EU-Rent Car Rental company, well-known in the Business Rules domain.**

This work was published as a technical communications paper at the International Conference on Logic Programming in [Van Hertum et al., 2013b] and as a technical report in [Van Hertum et al., 2013a]. A abstract was published in [Van Hertum, 2014]

- Imagine a setting where there are a set of principles and a set of resources, and a set of agents that together have to decide which principal gets access to which resource. Some agents own certain resources for which they decide whether to give access or to delegate that decision to others. To model this setting in a logic, a logic is needed where each agent can have its own knowledge base and they can refer to each other's knowledge. To be able to delegate decisions to others, an agent has to know what the other agents know. To delegate the decision to revoke to other agents, an agent has to know what other agents do not know. When Moore first formulated autoepistemic logic (AEL) in [Moore, 1984] this was motivated by the idea to consider an agent's theory to be a complete characterization of what the agent knows. With a similar motivation we introduce dAEL: distributed autoepistemic logic. dAEL is a multi-agent generalisation of AEL, where the agents have full introspection into each other's knowledge (or stated differently, where the agents refer to each others public belief). This assumption is motivated by applications in access control. We show the applications and compare this logic to other formalisms.

  Summarized: **dAEL is a multi-agent generalisation of autoepistemic logic that we developed, motivated by an application in Access Control. It assumes a setting where the agents are open, honest and fully collaborative.**

  This work was published as a conference paper at the International Joint Conference on Artificial Intelligence in [Van Hertum et al., 2016a].

- When reasoning about agents, sometimes we want to state that a certain knowledge base is all that an agent knows. Levesque captured this idea using “Only Knowing” [Levesque, 1990]. Only knowing a formula φ (denoted as $O_A\varphi$) by an agent A implies that A knows φ (denoted as $K_A\varphi$), and that for every formula ψ that is not a logical consequence of φ, A does not know ψ ($\neg K_A\psi$). Multi-agent only knowing has been studied before [Halpern and Lakemeyer, 2001, Belle and Lakemeyer, 2010], and the combination of common knowledge and only knowing in a multi-agent context was first discussed in [Belle and Lakemeyer, 2015]. We propose a new semantical structure that has a clean mathematical structure, motivated by issues we discovered in the earlier approaches. We formalize public announcements in this logic and show how the muddy children puzzle can be modelled in the proposed logic.

  Summarized: **We propose a new semantical structure for a logic accommodating operators for knowledge, only and common knowledge.**

1.1 Structure of the text

This text is structured as follows:

- Chapter 2 introduces background knowledge: concepts and notation not part of this research but used throughout the text. It introduces first-order logic, its extensions in FO($\cdot$), the knowledge base paradigm, autoepistemic logic and approximation fixpoint theory.

- Chapter 3 studies the interactive configuration application. It introduces the field of interactive configuration, studies which tasks are used in a configuration application and how they can be formalize as logical inferences.

- Chapter 4 handles the Business Rules application. In order to compare rule-based systems to our knowledge base paradigm we study how to model use cases of the EU-Rent Car Rental company in our KBS. We propose new inferences and language extensions to make this possible.

- Chapter 5 introduces dAEL: distributed autoepistemic logic. dAEL is a generalisation of autoepistemic logic for a distributed setting. Different agents can each have their own theories and refer to each other’s knowledge. It is motivated by an application in access control and assumes completely open, honest and communicative agents.

- Chapter 6 proposes $\mathcal{COL}_m$: a multi-agent modal logic accommodating knowledge, common knowledge and only knowing. We propose a new semantical structure, show the issues with previous approaches and motivate why this semantic structure does not have these issues. This logic makes no assumptions on introspection or truthfulness.
- Chapter 7 presents the conclusions of this research. We also give some directions for possible future work.

2 Background

This preliminary chapter looks at concepts and notations, not part of this research, but needed and used throughout this text. It starts with a look at First-order Logic (FO) (or classical logic), a formal language for which we formulate new extensions in this thesis. Afterwards, we look at some extensions that have already been formalized in the FO$(\cdot)$ family of languages.

We introduce the Knowledge Base Paradigm, and its implementation IDP. We present the ideology of the paradigm and give an overview of the supported reasoning tasks, or inferences.

In order to define some of the extensions, we use an algebraical framework: AFT. The needed concepts will be introduced in Section 2.6. This chapter will be concluded with a section containing a short introduction to the different semantics of autoepistemic logic (AEL), defined by AFT.

2.1 First-order Logic

This section only gives an overview of FO and the notations that will be used through the rest of this thesis. For a more detailed exposition, see for example [Enderton, 2001].

To define a formal language, we need to look at a number of aspects of that language. First, a set of symbols that can be used in that language is needed, which we call a *vocabulary*. Given this set of symbols, we need rules that define how these can be used to form correct sentences of the language: the *syntax*. Lastly, we need a *semantics*. The semantics of a language is a mathematical account of what sentences of that language mean.

To illustrate all concepts and notation, we use a running example: the planning of a course schedule for a high school, involving courses, teachers and groups of students.

Example 2.1.1. We look at a course scheduling application that schedules classes for teachers, courses and groups of students for one day, that satisfy following constraints:

- A teacher can only be teaching one class at a time.
- A group of students needs to have at least one class for each course.
- A teacher can only teach the classes he is qualified for.

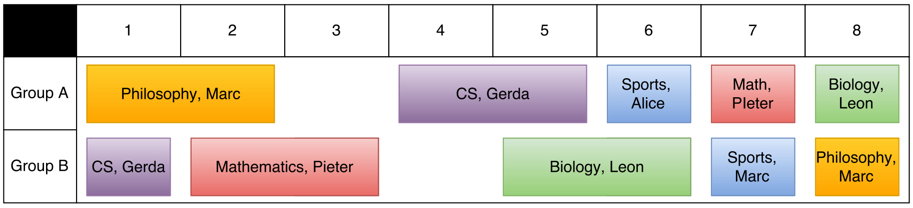

Figure 2.1: An example schedule for Example 2.1.1

2.1.1 Syntax and Semantics

FO uses two types of symbols: *logical* and *non-logical* symbols. The logical symbols have a fixed meaning in the language, while the non-logical symbols are the context-dependent symbols that are declared in the vocabulary. The set of logical symbols of FO contains variables $(x, y, z, \ldots)$, logical connectives ('$\neg$', '$\land$', '$\lor$', $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$), quantifiers ('$\forall$','$\exists$') and punctuation symbols ('(', ')', '.').

A vocabulary Σ declares the non-logical symbols, and consists of:

- A set of predicate symbols: Σ_P
- A set of function symbols: Σ_F

We say a vocabulary Σ' is a *subvocabulary* of a vocabulary Σ, if $\Sigma'_P \subseteq \Sigma_P$ and $\Sigma'_F \subseteq \Sigma_F$. With each predicate/function symbol σ, we associate a natural number, its arity, which indicates the number or arguments the symbol takes. We sometimes write σ/n to denote that σ is a predicate with arity n and $\sigma/n:$ to denote it is a function with arity n. Propositional symbols are predicates of arity 0 and constants are 0-ary function symbols. Propositional symbols also include $\top$ and $\bot$, denoting true and false.

Example 2.1.2. For our course scheduling domain, we have the following vocabulary:

$$\begin{aligned}
&\Sigma = \{\\
&\quad \Sigma_P = \{Time/1,\ Course/1,\ Teacher/1,\ Group/1,\ Class/4\}\\
&\quad \Sigma_F = \{Qualification/1:\}\\
&\}
\end{aligned}$$

In FO, we use a *structure* to define the semantics. A Σ-structure S associates values to the predicate and function symbols in Σ.

Definition 2.1.3. A Σ-structure S consists of

- The domain or universe D_S. This is the set of all values we consider, which are called the *domain elements*.
- For each predicate P/n, S contains an interpretation P^S: a subset of $(D_S)^n$.
- For each function f/n, S contains an interpretation f^S: a mapping $f^S : (D_S)^n \to D$. Sometimes we also identify an interpretation f^S with a subset of $(D_S)^{n+1}$.

We assume the predicate symbol $=/2$ (equality) to be implicitly contained in every vocabulary and interpreted by every structure in the standard way.

Example 2.1.4. A structure S for vocabulary Σ for course scheduling can be:

$$\begin{aligned}
&S = \{\\
&\quad D_S = \{G_A, G_B, Pieter, Gerda, Marc, Math, CS, Philosophy, 1..8\}\\
&\quad Time^S = \{1..8\}\\
&\quad Course^S = \{Math, CS, Philosophy\}\\
&\quad Teacher^S = \{Pieter, Gerda, Marc\}\\
&\quad Group^S = \{G_A, G_B\}\\
&\quad Class^S = \{(1, Marc, Philosophy, G_A), (4, Marc, Philosophy, G_B),\\
&\qquad (1, Gerda, CS, G_B), (3, Pieter, Math, G_A),\\
&\qquad (4, Gerda, CS, G_A), (7, Pieter, Math, G_B), (8, Pieter, Math, G_B)\}\\
&\quad Qualification^S = \{Marc \to Philosophy,\ Gerda \to CS,\\
&\qquad Pieter \to Math\}\\
&\}
\end{aligned}$$

Where $(1, Marc, Philosophy, G_A)$ for example means that Marc teaches Philosophy to group A in the first hour of the day.

Using the symbols of a vocabulary, we construct *terms* and *formulas* in FO.

Definition 2.1.5. Given a vocabulary Σ, the set of *terms* $\mathbb{T}$ over Σ is inductively defined as:

1. Each variable is a term.
2. If $t_1, \ldots t_n$ are terms, and $f/n \in \Sigma_F$, then $f(t_1, \ldots, t_n)$ is a term.

We usually consider logical expressions in the context of a fixed domain D, in which case domain elements are also considered terms.

Definition 2.1.6. Given a vocabulary Σ, a *formula* over Σ is inductively defined as:

1. If $t_1, \ldots t_n$ are terms, and $P/n \in \Sigma_P$, then $P(t_1, \ldots, t_n)$ is a formula, called an *atom*.
2. If φ and ψ are formulas, and x is a variable, then $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\exists x \varphi$ and $\forall x \psi$ are formulas.

We assume that $\neg$ binds stronger than $\wedge$ and $\vee$, which bind stronger than $\forall$ and $\exists$. Expressions $\varphi \Rightarrow \psi$, $\varphi \Leftarrow \psi$ and $\varphi \Leftrightarrow \psi$ are shorthand for $\neg\varphi \vee \psi$, $\varphi \vee \neg\psi$ and respectively $(\neg\varphi \vee \psi) \wedge (\varphi \vee \neg\psi)$. As is standard in mathematics, we often add parenthesis to improve readability. We use $\bar{t}$ to denote tuples of terms. If

$\bar{x}$ denotes the tuple of variables $(x_1, \ldots, x_n)$, then $(\forall \bar{x} \varphi)$ denotes the formula $\forall x_1 \forall x_2 \ldots \forall x_n \varphi$, likewise for $(\exists \bar{x} \varphi)$. We also denote the language of first-order logic by $\mathcal{L}_{FO}$.

The subset of the language of first-order logic that uses no quantification ($\forall$, $\exists$) and only propositional symbols, is called *propositional logic*, sometimes denoted as $\mathcal{L}_{Prop}$.

A *literal* is an atom $P(\bar{t})$ (a positive literal) or the negation $\neg P(\bar{t})$ of an atom (a negative literal). In a formula $\forall x \varphi$ or $\exists x \varphi$, we call φ the scope of x. Every occurrence of x in its scope is said to be bound. A variable is bound if all of its occurrences are bound. Non-bound variables are called free. A sentence is a formula with no free variables.

Definition 2.1.7. Given a vocabulary Σ: a Σ-theory T is a set of sentences over Σ.

Example 2.1.8. In the introduction of this running example (Example 2.1.1), we stated the following constraints:

1. A teacher can only be teaching one class at a time.
2. A group of students needs to have at least one class for each course.
3. A teacher can only teach the classes he is qualified for.

We translate them to FO:

1. $(\forall ti,\ te,\ c_1,\ c_2,\ g_1,\ g_2)(Class(ti, te, c_1, g_1) \wedge Class(ti, te, c_2, g_2) \Rightarrow c_1 = c_2 \wedge g_1 = g_2)$.
2. $(\forall g,\ c)(Group(g) \wedge Course(c) \Rightarrow (\exists ti,\ te)(Class(ti, te, g, c)))$.
3. $(\forall ti,\ te,\ c,\ g)(Class(ti, te, c, g) \Rightarrow Qualification(te) = c)$.

2.1.2 Semantics

For a given Σ-structure S, we define when a formula (and by extension a theory, as set of sentences) is satisfied in S.

Definition 2.1.9. We define the truth assignment function of a structure S on a formula φ by structural induction on φ:

- $P(\bar{t})^S$ is true iff $\bar{t}^S \in P^S$, and false otherwise,

- $\neg\psi^S$ is true iff ψ^S is false, and false otherwise,
- $(\varphi \wedge \psi)^S$ is true iff ψ^S is true and φ^S is true, and false otherwise,
- $(\varphi \vee \psi)^S$ is true iff ψ^S is true or φ^S is true, and false otherwise,
- $\forall x : \psi$ is true iff for each $d \in D^S$, $\psi[x/d]^S$ is true, and false otherwise,
- $\exists x : \psi$ is true iff for at least one $d \in D^S$, $\psi[x/d]^S$ is true, and false otherwise.

If φ^S is true, we also say that S satisfies φ, or S is a model of φ, with notation $S \models \varphi$. A structure is a model of a theory T if it satisfies all sentences in T.

2.2 Partial structures

The structure as defined in Definition 2.1.3 gives an account of the state of the world. However, often we have only partial information on the world. We define partial three-valued structures, as a generalization of structures.

A *partial set* on a domain D is a function from D to $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. It is called a partial set, because a domain element can now be *in* the set ($\mathbf{t}$), *not in* the set ($\mathbf{f}$) or *unknown* to be in the set ($\mathbf{u}$). A partial set is two-valued (or total) if $\mathbf{u}$ does not belong to its range. A *partial structure* $\mathcal{S}$ [1] consists of a domain D and an assignment of a partial set $\sigma^{\mathcal{S}}$ to each predicate or function symbol $\sigma \in (\Sigma_P \cup \Sigma_F)$, called the *interpretation* of σ in $\mathcal{S}$. For a predicate p with arity n, the interpretation $p^{\mathcal{S}}$ of p in $\mathcal{S}$ is a partial set on domain D^n, and the interpretation $f^{\mathcal{S}}$ of a function f with arity n in $\mathcal{S}$ is a partial set on domain D^{n+1}. In case the interpretation of (a predicate or function symbol) σ in $\mathcal{S}$ is a two-valued set, we abuse notation and use $\sigma^{\mathcal{S}}$ as shorthand for $\{\overline{d} | \sigma^{\mathcal{S}}(\overline{d}) = \mathbf{t}\}$, otherwise we use following shorthands

$$\sigma^{\mathcal{S}}_{\mathbf{t}} = \{\overline{d} | \sigma^{\mathcal{S}}(\overline{d}) = \mathbf{t}\}$$

$$\sigma^{\mathcal{S}}_{\mathbf{f}} = \{\overline{d} | \sigma^{\mathcal{S}}(\overline{d}) = \mathbf{f}\}$$

$$\sigma^{\mathcal{S}}_{\mathbf{u}} = \{\overline{d} | \sigma^{\mathcal{S}}(\overline{d}) = \mathbf{u}\}$$

The truth order $>_t$ on truth values is defined by $\mathbf{t} >_t \mathbf{u} >_t \mathbf{f}$. The (partial) precision order $>_p$ on truth values is defined by $\mathbf{u} <_p \mathbf{t}$ and $\mathbf{u} <_p \mathbf{f}$.

[1] Note the difference in typography between a partial structure $\mathcal{S}$ and a total structure S.

To extend the valuation function as in Definition 2.1.9, we use the Kleene semantics [Kleene, 1952], which extends the truth tables of all operators to handle unknown. E.g., $\varphi \lor \psi$ is true if either φ or ψ is true, false if both are false and unknown otherwise.

The precision $\leq_p$ order can be extended pointwise to partial sets and partial structures, denoted $\mathcal{S} \leq_p \mathcal{S}'$. This means that an interpretation has become more precise if tuples of domain elements that were previously mapped to unknown now map to true or false. Similar for the truth order $\leq_t$. Notice that total structures are the maximally precise ones in the precision order. We will illustrate the extended precision relation in Example 2.2.1. When $\mathcal{S} \leq_p \mathcal{S}'$, we say that $\mathcal{S}'$ extends $\mathcal{S}$, that $\mathcal{S}$ approximates $\mathcal{S}'$ or that $\mathcal{S}'$ is an expansion of $\mathcal{S}$. The satisfaction relation is monotone: if $\mathcal{S} \leq_p \mathcal{S}'$, then $\varphi^{\mathcal{S}} \leq_p \varphi^{\mathcal{S}'}$. Hence, if a formula is true in a partial structure, it is true in all two-valued expansions of it. For a theory T and a partial structure $\mathcal{S}$, we say that $\mathcal{S}$ is a model of T, or $\mathcal{S}$ satisfies T (in symbols $\mathcal{S} \vDash T$) if $T^{\mathcal{S}} = \mathbf{t}$ and $\mathcal{S}$ is two-valued. We sometimes abuse notation and write $T \vDash \varphi$ (or $T \models T'$) for the entailment relation, as a shorthand for "For every structure S such that $S \vDash T$, we have $S \vDash \varphi$.".

A total structure S is called *functionally consistent* if for each function f, the interpretation f^S is the graph of a function $D^n \rightarrow D$. A partial structure $\mathcal{S}$ is functionally consistent if it has a functionally consistent two-valued extension. Unless stated otherwise, we assume for the rest of this thesis that all (partial) structures are functionally consistent.

A domain atom (domain term) is a tuple of a predicate symbol P (a function symbol F) and a tuple of domain elements $(d_1, \dots, d_n)$. We will denote it as $P(d_1, \dots, d_n)$ (respectively $F(d_1, \dots, d_n)$). We say a domain term t is uninterpreted in $\mathcal{S}$ if $\{d | d \in D \land (t = d)^{\mathcal{S}} = \mathbf{u}\}$ is non-empty.

Example 2.2.1. To represent input information for Example 2.1.1, such as the teachers, groups and courses that have to be planned and some constraints on when a certain teacher is not available, we use a partial structure $\mathcal{S}'$. $\mathcal{S}'$ is a partial structure that approximates S (Example 2.1.4). With

$$
\begin{aligned}
&\mathcal{S}' = \{\\
&\quad D_S = \{G_A, G_B, Pieter, Gerda, Marc, Math, CS, Philosophy, 1..8\}\\
&\quad Time^S = \{1..8\}\\
&\quad Course^S = \{Math, CS, Philosophy\}\\
&\quad Teacher^S = \{Pieter, Gerda, Marc\}\\
&\quad Group^S = \{G_A, G_B\}\\
&\quad Class^S_{\mathbf{t}} = \{(1, Marc, Philosophy, G_A), (4, Marc, Philosophy, G_B)\}\\
&\quad Class^S_{\mathbf{f}} = \{(5, Gerda, CS, G_A), (5, Gerda, CS, G_B),\\
&\qquad\qquad (8, Gerda, CS, G_A)\}\\
&\}
\end{aligned}
$$

An associated theory

The associated theory $T_{\mathcal{S}}$ of a partial structure $\mathcal{S}$ is a representation of the information contained in $\mathcal{S}$ as a theory, which will be used in Chapter 3. It is defined by the following collection of constraints. For every predicate symbol P, this collection contains two sets of constraints:

$$\{P(\overline{d})|\overline{d} \in P^{\mathcal{S}}_{\mathbf{t}}\}$$

$$\{\neg P(\overline{d})|\overline{d} \in P^{\mathcal{S}}_{\mathbf{f}}\}$$

and two sets of constraints for every function symbol F:

$$\{F(\overline{d}) = e|(\overline{d}, e) \in F^{\mathcal{S}}_{\mathbf{t}}\}$$

$$\{\neg F(\overline{d}) = e|(\overline{d}, e) \in F^{\mathcal{S}}_{\mathbf{f}}\}$$

Given a partial structure $\mathcal{S}$, the domain structure $\mathcal{S}_D$ is the structure containing only the domain of $\mathcal{S}$. It is easy to see that $\mathcal{S}$ contains the same information as $T_{\mathcal{S}} \cup \mathcal{S}_D$.

2.3 Extensions for first order logic: FO(·)

In this section, we discuss a number of extensions for FO. As is standard in literature, we denote an extension to FO as $FO(x)$, with x an identifier for that extension. FO(·) [Denecker and Ternovska, 2008] is used to denote the set of extensions for FO. Abusing notation we also use FO(·) as a shorthand for the language that is the union of all extensions discussed below.

2.3.1 Sorted logic: FO(Types)

Above, we defined one-sorted (or *untyped*) FO, which used a domain D in any interpretation. *Many-sorted* (or *typed* logic (FO(Types)) is an extension that categorizes domain elements into different sorts or types. For many applications, typed FO is a more natural language. In FO(Types), the vocabulary contains in addition to a set of predicates Σ_P and a set of functions Σ_F now also a set of types Σ_T. A predicate of arity n now has a type $[\tau_1, \dots, \tau_n]$, which is an n-tuple of type symbols $\tau_i \in \Sigma_T$. A function of arity n now has a type $[\tau_1, \dots, \tau_n] \rightarrow \tau_{n+1}$, a $(n+1)$-tuple of type symbols[2]. The definition of a structure is adapted as follows:

[2]A predicate P with type $[\tau_1, \dots, \tau_n]$ is also denoted as $P(\tau_1, \dots, \tau_n)$. A function is denoted $f(\tau_1, \dots, \tau_n) : \tau$.

- Instead of a domain D, an interpretation contains a domain D_τ for every $\tau \in \Sigma_T$.
- A (partial) interpretation of a predicate $P[\tau_1, \dots \tau_n]$ is a (partial) subset of $D_{\tau_1} \times D_{\tau_2} \times \dots \times D_{\tau_n}$.
- A (partial) interpretation of a function $P[\tau_1, \dots \tau_n] \to \tau_{n+1}$ is a (partial) subset of $D_{\tau_1} \times D_{\tau_2} \times \dots \times D_{\tau_n} \times D_{\tau_{n+1}}$.

Throughout this thesis, we implicitly always use typed logics, while in theorems and proofs we use the untyped variant for readability. We can do this, since the result of Oberschelp [Oberschelp, 1962] shows that typed FO can be reduced to untyped FO by introducing a predicate of arity 1, for each sort in the vocabulary of the typed logic.

Types are intended to mimic the classification in a domain. As such, it is natural to extend the notion of a type to *type hierarchies*: a type can be a subtype or supertype of another type, indicating that an interpretation of the former will be a subset/superset of the latter.

Example 2.3.1. A typed vocabulary for Example 2.1.1 can look like:

$$\begin{aligned}
&\Sigma = \{\\
&\qquad \Sigma_T = \{Time,\ Course,\ Teacher,\ Group\}\\
&\qquad \Sigma_P = \{Class(Time, Teacher, Course, Group)\}\\
&\qquad \Sigma_F = \{\mathit{Qualification}(Teacher) : Course\}\\
&\}
\end{aligned}$$

Note that we have less predicates in this vocabulary then before. Because of the types, there is no need anymore for unary predicates representing the set of teachers, groups, ...

2.3.2 Partial functions: FO(PF)

The functions used in FO are *total* functions. A function of arity n has an image for every tuple of length n in the domain of the function. In real life and mathematics, we regularly come across functions that are *partial*. A good example is the division of integers: $\div : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Q}$, since there is no image for any tuple of the form $(x, 0)$, with $x \in \mathbb{Z}$. A partial function has for each input tuple, *zero* or *one* image(s). In context of a three-valued structure $\mathcal{S}$, a partial function f should, given an input tuple $\overline{d}$, have at most one tuple $(\overline{d}, d)$ in $f^{\mathcal{S}}_{\mathbf{t}}$.

If f/n is a partial function, we allow that for certain $\overline{d} \in D^n$, $f(\overline{d})$ is undefined. We recursively extend the interpretation of terms and formulas as follows. A term that has an undefined subterm is also undefined (for example a term $+(3, \div(5,0))$ would be undefined) and an atom containing an undefined term is *false*.

2.3.3 Sets and aggregates: FO(Agg)

In FO(Agg), we define a *set expression* as an expression of the form $\{\overline{x} : \varphi\}$ or of the form $\{\overline{x} : \varphi : t\}$, with $\overline{x}$ a tuple of variables, φ a formula in FO and t a term as defined in FO. Call $\overline{y}$ the free variables of φ, then $\overline{x} \subseteq \overline{y}$ and define $\overline{z} = \overline{y} \setminus \overline{x}$, the free variables of the set expression. Given a structure S with domain D and an assignment $\overline{d}$ to the free variables $\overline{z}$, the interpretation $\{\overline{x} : \varphi\}^S$ is the set $\{\overline{d}' \in \bar{D} | \varphi[\overline{x}/\overline{d}', \overline{z}/\overline{d}]^S = \mathbf{t}\}$ and $\{\overline{x} : \varphi : t\}^S$ is the multiset $\{t[\overline{x}/\overline{d}', \overline{z}/\overline{d}]^S | \overline{d}' \in \bar{D}\}$ and $\varphi[\overline{x}/\overline{d}', \overline{z}/\overline{d}]^S = \mathbf{t}\}$.

An *aggregate term* is a term of the form $Agg(E)$, with Agg an aggregate function symbol and E a set expression as defined above. At this point *Max, Min, Sum, Card* and *Prod* are defined in the FO(Agg) language. The $Card$ aggregate maps a set expression $\{\overline{x} : \varphi\}$ to the number of elements that set contains. The Max, Min, Sum and $Prod$ aggregates map a set expression $\{\overline{x} : \varphi : t\}$ to respectively the maximum, minimum, sum and product of the terms t of the tuples $(\overline{x}, t)$ that satisfy φ. For set expressions $\{\overline{x} : \varphi\}$, the Max, Min, Sum and $Prod$ aggregates are only defined for sets that contain tuples of length one, and map such a set expression E to respectively the maximum, minimum, sum and product of the first (and only) element of each tuple in that set. By consequence, it makes no sense using this operators on sets containing tuples of arity higher then 1. Sum and product aggregates are undefined for sets that contain non-numeric values and the minimum and maximum of an empty set is also undefined. Note that this means that those four aggregates are partial functions.

We also use aggregates for another extension we use in this text: extended quantifications. Since they are just syntactic sugar, we shortly introduce them here. We introduce $(\exists_{\sim i} \overline{x})\varphi$, with $\sim \in \{=, <, >, \leq, \geq\}$ as a shorthand for:

$$Card\{\overline{x} : \varphi\} \sim i$$

2.3.4 Definitions: FO(ID)

When we have a graph, with nodes and edges, there is an important concept, that is inexpressible in FO: the reachability between nodes [Libkin, 2004] (or, put differently the transitive closure in a graph). Say we have a vocabulary containing a type $Node$ and predicates $Edge(Node, Node)$ and $Reach(Node, Node)$, there exists no formula φ in FO such that in any model M satisfying φ it is the case that $(n_1, n_2) \in Reach^M$ iff there is a path between n_1 and n_2 in the graph.

It is however not a difficult concept to formalize, using a recursive or inductive definition:

- Node n_2 is reachable from node n_1 if there is an edge connecting them;
- Node n_2 is reachable from node n_1 if there is a node n that is reachable from n_1 such that n_2 is reachable from n.

In this section we present FO(ID) [Denecker, 2000]: an extension that allows to express (inductive) definitions over FO formulas.

Definitions Δ are sets of rules δ of the form $\forall \overline{x} : P(\overline{t}) \leftarrow \varphi$, with the free variables of φ among the variables in $\overline{x}$ and φ a first-order formula. We call $P(\overline{t})$ the head ($head(\delta)$) and φ the body ($body(\delta)$) of the rule. We say that Δ *defines* Q if Δ contains a rule δ with $head(\delta) = Q(\overline{t})$. We use $Def(\Delta)$ to denote all symbols defined in Δ; all other symbols are called *parameters*, or *opens*, denoted as $Open(\Delta)$. FO(ID) is now defined as the language, where a formula is a FO formula or a definition Δ.

Example 2.3.2. To show how a definition looks, we formalize the definition of reachability in FO(ID). We take the vocabulary $\Sigma = \{Reach/2, Edge/2\}$ and show a definition Δ that defines $Reach$ in terms of $Edge$.

$$\left\{ \begin{array}{l} \forall x\ y : Reach(x, y) \leftarrow Edge(x, y). \\ \forall x\ y : Reach(x, y) \leftarrow \exists z : Reach(x, z) \land Reach(z, y). \end{array} \right\}$$

The satisfaction relation for FO can be extended for FO(ID), using the well-founded semantics [Van Gelder, 1993]. This semantics formalizes the informal semantics of rule sets as inductive definitions [Denecker, 1998, Denecker et al., 2001, Denecker and Vennekens, 2014]. Let Δ' be the definition constructed from Δ by replacing each rule $\forall \overline{x} : P(\overline{t}) \leftarrow \varphi$ with $\forall \overline{y} : P(\overline{y}) \leftarrow \exists \overline{x} : \overline{t} = \overline{y} \land \varphi$. The interpretation I satisfies Δ ($I \models \Delta$) if I is a parametrized well-founded model of Δ, that means that I is the well-founded model of Δ' when the open symbols are interpreted as in I.

A definition Δ is *total* if for each structure I that is two-valued on $Open(\Delta)$ and is unknown on $Def(\Delta)$, a two-valued expansion of I exists which is a model of Δ [Denecker and Ternovska, 2008]. The *completion* of Δ for a symbol P, defined in Δ by the rules $\forall \overline{x}_i : P(\overline{t}_i) \leftarrow \varphi_i$ with $i \in [1, n]$, is the set consisting of the sentence $\forall \overline{x}_i : \varphi_i \Rightarrow P(\overline{t}_i)$ for each $i \in [1, n]$ and the sentence

$\forall \overline{x} : P(\overline{x}) \Rightarrow \bigvee_{i\in[1,n]}(\overline{x} = \overline{t}_i \wedge \varphi_i)$. This set is denoted as $\mathrm{comp}(P, \Delta)$, the union of all these sets for Δ as $\mathrm{comp}(\Delta)$. It is well-known that, if $I \models \Delta$, then $I \models \mathrm{comp}(\Delta)$ but not always vice-versa (e.g., the inductive definition expressing transitive closure is stronger than its completion).

2.3.5 Time-dependent specifications in FO(·): LTC

Further in this thesis, we look at specification of processes, and as such have an FO(·) theory that has a *dynamic domain*. There exist many languages, suited for this purpose, such as situation calculus [Reiter, 2001], event calculus [Shanahan, 1997] and a family of action languages [Gelfond and Lifschitz, 1998]. In this section, we define a specific set of FO(·) vocabularies and theories, that make up such a calculus: Linear Time Calculus (LTC). This section only contains a short introduction on the subject, explaining the results that will be used in this thesis. For a more in-depth discussion on the subject, we refer to [Bogaerts et al., 2014a].

We first define a linear-time vocabulary Σ_{LTC} and a linear-time structure S_{LTC} (a specific case of a vocabulary in FO(·)).

Definition 2.3.3. A linear-time vocabulary is a typed first-order vocabulary Σ_{LTC} such that:

- Σ_{LTC} has a type $Time$, a constant $Z : Time$ and a function $Next(Time) : Time$.
- All other symbols in Σ_{LTC} have at most one argument of type $Time$.
- Apart from Z and $Next$, the type of the image of functions is not $Time$.

We partition symbols in Σ_{LTC} in three categories:

1. $Time$, Z and $Next$ are LTC-symbols
2. Non-LTC-symbols with a $Time$ argument are *dynamic symbols*.
3. Non-LTC-symbols without a $Time$ argument are *static symbols*.

For ease of notation, we assume that $Time$ always occurs last in dynamic symbols. Intuitively, the Z constant represents 0, and the $Next$ function represents the successor function.

Definition 2.3.4. A (partial) linear-time structure $\mathcal{S}_{LTC}$ is a (partial) structure over a vocabulary Σ_{LTC} such that

- *Time* is interpreted by a subset of the natural numbers $\mathbb{N}$.
- *Next* is interpreted by the successor function $n \mapsto n+1$ for the natural numbers.
- Z is interpreted by 0.

Definition 2.3.5. If $P(\tau_1, \ldots, \tau_{n-1}, Time)$ is a dynamic predicate symbol in Σ_{LTC}, then we define $P_{ss}(\tau_1, \ldots, \tau_{n-1})$ as its *projected symbol.* Analogously: $f_{ss}(\tau_1, \ldots, \tau_{n-1}) : \tau$ is the projected symbol of a dynamic function symbol $f_{ss}(\tau_1, \ldots, \tau_{n-1}, Time) : \tau$ in Σ_{LTC}.

Definition 2.3.6. Given a Σ_{LTC}, we define the derived *single state* vocabulary Σ_{LTC}^{ss} as:

- The single state vocabulary contains all types of Σ_{LTC}, except *Time*.
- The single state vocabulary contains all static symbols of Σ_{LTC}.
- For each dynamic symbol σ, the single state vocabulary contains its projection σ_{ss}.

Intuitively a structure over Σ_{LTC}^{ss} describes one single state in the process that one can describe over Σ_{LTC}.

We now define a mapping for structures over Σ_{LTC} to single state structures over Σ_{LTC}^{ss}.

Definition 2.3.7. Let Ω be the interpretation of a dynamic symbol σ in a structure $\mathcal{S}$ over Σ_{LTC} and $k \in \mathbb{N}$. The set of tuples $(d_1, \ldots, d_{n-1})$ such that $(d_1, \ldots, d_{n-1}, k) \in \Omega$ is the *k-projection* of Ω, denoted $\pi_k(\Omega)$. It is a mapping from a structure representing the entire process to a representation of the state of affairs on timepoint k.

Given a linear-time structure $\mathcal{S}$ over Σ_{LTC}, we define its projection on time point i, $i \in Time^{\mathcal{S}}$ as $\mathcal{S}^i$ as follows:

- For each type $\tau \in \Sigma_{LTC}$, $\tau \neq Time$: $\tau^{\mathcal{S}^i} = \tau^{\mathcal{S}}$.
- For each static symbol $\sigma \in \Sigma_{LTC}$: $\sigma^{\mathcal{S}^i} = \sigma^{\mathcal{S}}$.
- For each dynamic symbol $\sigma \in \Sigma_{LTC}$: $\sigma^{\mathcal{S}^i} = \pi_i(\sigma^{\mathcal{S}})$.

We finally define the reverse.

Definition 2.3.8. Given a linear-time vocabulary Σ_{LTC}, its single state vocabulary Σ_{LTC}^{ss} and two structures S_0 and S_1 over Σ_{LTC}^{ss} such that

- For each type $\tau \in \Sigma_{LTC}$, $\tau \neq Time$: $\tau^{\mathcal{S}_0} = \tau^{\mathcal{S}_1}$.
- For each static symbol $\sigma \in \Sigma_{LTC}$: $\sigma^{\mathcal{S}_0} = \sigma^{\mathcal{S}_1}$.

We define the bistate structure $S_{bs} = bistate(\Sigma_{LTC}, S_0, S_1)$ as follows:

- $Time^{S_{bs}} = \{0, 1\}$
- For each type $\tau \in \Sigma_{LTC}$, $\tau \neq Time$: $\tau^{\mathcal{S}_i} = \tau^{\mathcal{S}}$ for each $i \in \{0, 1\}$.
- For each static symbol $\sigma \in \Sigma_{LTC}$: $\sigma^{\mathcal{S}_i} = \sigma^{\mathcal{S}}$ for each $i \in \{0, 1\}$.
- For each dynamic symbol $\sigma \in \Sigma_{LTC}$: $\sigma^{S_{bs}}$ is the interpretation such that $\pi_0(\sigma^{S_{bs}}) = \sigma^{S_0}$ and $\pi_1(\sigma^{S_{bs}}) = \sigma^{S_1}$.

2.4 The Knowledge Base Paradigm

The key idea in the KB paradigm lies in applying a strict separation of concerns to information and to problem solving. Instead of compiling knowledge to working imperative code, a user states the domain knowledge explicitly in a formal language in the KB paradigm and uses predefined reasoning engines to get the desired behaviour from this. A Knowledge Base System (KBS) [Denecker and Vennekens, 2008] allows information to be stored in a knowledge base, and provides a range of inference methods. With these inference methods various types of problems and tasks can be solved using the *same knowledge base*. As such the knowledge base is neither a program nor a description of a problem, it cannot be executed or run. It is nothing but information. However, this information can be used to solve multiple sorts of problems. Many declarative problem solving paradigms are mono-inferential: they are based on one form of inference. In comparison, the KB paradigm is multi-inferential. We believe that this implements a more natural, pure view of what declarative logic is aimed to be. The FO$(\cdot)$ KB project is a research project that runs now for a number of years. Its aim is to integrate different useful language constructs and forms of inference from different declarative paradigms in one rich declarative language and a KBS. So far, it has led to the KB language FO$(\cdot)$ [Denecker and Ternovska, 2008] and the KB system IDP [De Cat et al., 2016].

2.4.1 The language of a KB system

Central in a KBS is a formal knowledge representation language, used for representing the domain knowledge. So the choice of a good language is very important. The language should be a rich language, such that users can represent a reasonable set of domain knowledge. It also should be natural, to reduce the chance for errors and improve readability. Lastly, a good Knowledge Representation (KR) language for a KBS should be modular, such that a specification of domain knowledge can be easily reused and extended.

Some literature states that a language should be not too expressive, such that “the language” remains decidable. In the context of our KB Paradigm, we do not support or believe in this statement, for two reasons. The first one is that there is no such thing as (un)decidability for a language, a specification in a language is pure information and when one is talking about being decidability, there is reasoning involved. Decidability can be defined for each task where a language is used for: for example, deduction for FO is undecidable, while model generation with a finite domain is decidable for FO. Second, while a more expressive language does indeed allow a user to express more difficult tasks, this does not imply that all tasks stated in a more expressive language become more difficult. On the contrary, using a more expressive statement allows the user to express structure in the problem (that can be exploited by the system) that would be lost in a lower level language.

In the IDP project, we have chosen for the language FO(Types, ID, Agg, PF), which (as stated above) we shorten as FO($\cdot$). The choice for FO as a base language is made because conjunction, disjunction, negation, and universal and existential quantification are natural (or one may even argue, the basic) connectives of information. They have a clear informal semantics that corresponds well to our intuitions. FO has, however, various shortcomings for which language extensions where defined. Types, because they impose a clear classification of objects into categories, reducing changes of mistakes while writing a specification. When quantifying in natural language, types are often involved: “Everyone is mortal”. Definitions have been added because FO cannot express inductively defined concepts. Definitions are often also very useful without induction: they express the definitional if, which is not the same as the material if, but in texts the word “if” is ambiguously used for both meanings. An example of a material if is the following statement: "If it rains, I will take my umbrella." In the situation where it does not rain, but it is very cloudy it is still reasonable of me to take my umbrella and the statement would not be a lie. This is a material implication: if the antecedent is false, the statement is true no matter the truth of the consequence. A definitional if does, by contrast, makes a statement about the situation in which the antecedent is false. For example: "I will join you in the bar, if my classes end on time". This sentence clearly implies that I will not join if my classes do not end on time.

With aggregates a user can express information on sets, reducing the need for very complex and long FO formulas (such as: “there exist at most 5... ”).

Partial functions are also omnipresent in natural language: “Which chess piece is on this square?”

2.4.2 Inferences in a KB system

In the KB paradigm, a specification is a bag of information. This information can be used for solving various problems by applying a suitable form of inference on it. In this section we give a short overview of inferences that are available in the IDP system. Further in this thesis we refer to these inferences as the *base inferences* of the IDP system.

Inferences are described as functions, taking a number of information building blocks (=(partial) structures, vocabularies, theories and terms) and returning an information building block. We assume a fixed vocabulary Σ and finite domain in all the structures. We recall the complexities, given in function of the domain size.

Modelexpand($T, \mathcal{S}$): input: theory T and partial structure $\mathcal{S}$; output: a model I of T such that $\mathcal{S} \leq_p I$ or $UNSAT$ if there is no such I. Modelexpand [Wittocx et al., 2008] is a generalization for FO($\cdot$) theories of the modelexpansion task as defined by Mitchell et al. [Mitchell and Ternovska, 2005]. Complexity of deciding the existence of a modelexpansion is in $\mathbf{NP}$.

Modelcheck(T, S): input: a total structure S and theory T over the vocabulary interpreted by S; output is the boolean value $S \models T$. Note that Modelcheck is a degenerate case of the Modelexpand inference, with $\mathcal{S}$ a total structure. Complexity is in $\mathbf{P}$.

Minimize($T, \mathcal{S}, t$): input: a theory T, a partial structure $\mathcal{S}$ and a term t of numerical type; output: a model $I \geq_p \mathcal{S}$ of T such that the value t^I of t is minimal. The term t represents a numerical expression whose value has to be minimized. This is an extension to the Modelexpand inference. The complexity of deciding that a certain t^I is minimal, is in $\mathbf{\Delta_2^P}$.

Propagate($T, \mathcal{S}$): input: theory T and partial structure $\mathcal{S}$; output: the most precise partial structure $\mathcal{S}_r$ such that for every model $I \geq_p \mathcal{S}$ of T it is true that $I \geq_p \mathcal{S}_r$. The complexity of deciding that a partial structure $\mathcal{S}'$ is $\mathcal{S}_r$ is in $\mathbf{\Delta_2^P}$. Note that we assume that all partial structures are functionally consistent, which implies that we also propagate functional integrity constraints.

Consequence(T_1, T_2) Input are two logical theories T_1 and T_2. Output is **true** if $T_1 \models T_2$. Consequence (or Deduction) is undecidable for FO($\cdot$).

Query($\mathcal{S}, E$): input: a (partial) structure $\mathcal{S}$ and a set expression $E = \{\overline{x} \mid \varphi(\overline{x})\}$; output: the set $A_Q = \{\overline{x} \mid \varphi(\overline{x})^{\mathcal{S}} = \mathbf{t}\}$. Complexity of deciding that a set A is A_Q is in **P**.

Progress(T, S_0) input: a single-state structure S_0 over Σ^{ss}_{LTC} and a theory T over a linear-time vocabulary Σ_{LTC} (Defintion 2.3.3); output: a single-state partial structure S_1 such that $bistate(\Sigma_{LTC}, S_0, S_1) \models T$. Complexity of deciding the existence of a progression is in **NP**.

Example 2.4.1. Within a course scheduling application, there are a number of tasks that all need the same information. A key task is to generate a schedule given a partial input structure that satisfies all constraints: a *model expansion* inference. In some cases all constraints cannot be satisfied because the teachers have too many constraints on their availability, then we might look for a schedule that has a minimal amount of conflicts, using the *minimisation* inference. Given a full schedule, the school might want to give teachers their personal schedule. For this, the school can use a *query* inference on the schedule.

There are many more examples of inferences that can be used in the context of a course scheduling application. In Chapter 3, we explore this multi-inferentiality for a bigger application: interactive configuration.

2.5 Autoepistemic Logic

The language $\mathcal{L}_k$ of AEL [Moore, 1985] is defined recursively using the standard rules for the syntax of first-order logic (Definition 2.1.6), augmented with:

$$K(\psi) \in \mathcal{L}_k \quad \text{if } \psi \in \mathcal{L}_k$$

Informally, the K operator is read as "it is known that". We call formulas in $\mathcal{L}_k$ that do not contain any occurrences of K *objective*, and we refer to an atom of the form $K\varphi$ as a *modal atom*. An AEL theory T is a set of formulas over $\mathcal{L}_k$. AEL uses the semantic concepts of standard modal logic. A *structure* is defined as usual in first-order logic. It formally represents a potential state of affairs of the world. We assume a domain D, shared by all structures, to be fixed throughout this section. In the context of modal logics, we sometimes also refer to structures as "worlds". Furthermore, we assume that for each $d \in D$, d is a constant symbol of $\mathcal{L}_k$ whose interpretation in all structures is d.

Definition 2.5.1. A Possible World Structure (PWS) Q is a set of structures.

It contains all structures that are consistent with an agent's knowledge. Possible world structures are ordered with respect to the amount of knowledge they contain.

Possible world structures that contain less structures possess more knowledge.

Definition 2.5.2. Given two PWS's Q_1 and Q_2, we define the *knowledge order* $\leq_K$ as:

$$Q_1 \leq_K Q_2 \text{ iff } Q_2 \subseteq Q_1$$

We say that Q_1 knows more than Q_2 if $Q_2 \leq_K Q_1$.

Note that the knowledge order is the inverse of the subset order. Following example will explain the intuitions about this:

Example 2.5.3. Say, we have a figure that has a color and a shape.

$$\begin{aligned}
&\Sigma = \{\\
&\quad \Sigma_F = \{color/0 :, shape/0 :\}\\
&\}
\end{aligned}$$

We define two structures:

$$\begin{aligned}
&S_1 = \{\\
&\quad D_{S_1} = \{blue, green, yellow, round, square\}\\
&\quad color^{S_1} = blue\\
&\quad shape^{S_1} = round\\
&\}\\
&S_2 = \{\\
&\quad D_{S_2} = \{blue, green, yellow, round, square\}\\
&\quad color^{S_2} = green\\
&\quad shape^{S_2} = round\\
&\}
\end{aligned}$$

If we now look at two PWS's: $Q = \{S_1\}$ and $Q' = \{S_1, S_2\}$. It is clear that $Q \subseteq Q'$, while Q contains more knowledge than Q'. Indeed, in Q the color of the figure is known to be *blue*, while in Q' it is only known to be *blue* or *green*.

The semantics of AEL is based on the S5 truth assignment. The *value* of a sentence $\varphi \in \mathcal{L}_k$ with respect to a possible world structure Q and a structure I (denoted $\varphi^{Q,I}$) is defined using the recursive rules for first-order logic augmented with:

$$(K\varphi)^{Q,I} = \mathbf{t} \qquad\qquad \text{if } \varphi^{Q,J} = \mathbf{t} \text{ for each } J \in Q.$$

Moore defines that Q is an *autoepistemic expansion* of T if for every world I, it holds that $I \in Q$ if and only if $T^{Q,I} = \mathbf{t}$.

The above definition is essentially a fixpoint characterisation. We define a underlying operator D_T, that maps Q to

$$D_T(Q) = \{I \mid T^{Q,I} = \mathbf{t}\}.$$

Autoepistemic expansions are exactly the fixpoints of D_T; they are the possible world structures that, according to Moore, express candidate belief states of an autoepistemic agent with knowledge base T.

Soon, researchers pointed out certain “anomalies” in the expansion semantics [Halpern and Moses, 1985, Konolige, 1988]. In the following years, many different semantics for AEL were proposed. It was only with the abstract algebraical framework *approximation fixpoint theory* (AFT) that a uniform view on those different semantics was obtained. We define several of the semantics of AEL later, after we introduce the algebraical preliminaries on AFT.

2.6 Approximation Fixpoint Theory

To define the semantics of some new extensions for FO in this thesis, we use AFT [Denecker et al., 2000], an algebraical framework that studies semantics of logics using approximators of valuations for that logic. In this section, we recall the basics of lattice theory and introduce the most important result of AFT that will be used in later chapters.

2.6.1 Sets, Lattices and Operators

A *complete lattice* $\langle L, \leq \rangle$ is a set L equipped with a partial order $\leq$, such that every set $S \subseteq L$ has both a least upper bound and a greatest lower bound, denoted $\mathrm{lub}(S)$ and $\mathrm{glb}(S)$. A complete lattice has a least element $\bot$ and a greatest element $\top$. As custom, we also use the notations $\bigwedge S = \mathrm{glb}(S)$, $x \wedge y = \mathrm{glb}(\{x,y\})$, $\bigvee S = \mathrm{lub}(S)$ and $x \vee y = \mathrm{lub}(\{x,y\})$. An operator $O : L \to L$ is *monotone* if $x \leq y$ implies that $O(x) \leq O(y)$. An element $x \in L$ is a *prefixpoint*, a *fixpoint*, a *postfixpoint* if $O(x) \leq x$, $O(x) = x$, respectively $x \leq O(x)$. Every monotone operator O in a complete lattice has a least fixpoint, denoted $\mathrm{lfp}(O)$, which is the limit (least upper bound) of the sequence

O^α given by:

$$O^0(x) = \bot$$

$$O^{\alpha+1} = O(O^\alpha(x))$$

$$O^\lambda(x) = \text{lub}(\{O^\alpha(x) \mid \alpha < \lambda\}), \text{ with } \lambda \text{ a limit ordinal}$$

2.6.2 AFT

Given a lattice L, AFT makes use of the bilattice L^2. We define *projections* for pairs as usual: $(x,y)_1 = x$ and $(x,y)_2 = y$. Pairs $(x,y) \in L^2$ are used to approximate all elements in the interval $[x,y] = \{z \mid x \leq z \wedge z \leq y\}$. We call $(x,y) \in L^2$ *consistent* if $[x,y]$ is non-empty and use L^c to denote the set of consistent elements. Elements $(x,x) \in L^c$ are called *exact*. We identify a point $x \in L$ with the exact bilattice point $(x,x) \in L^c$. The *precision order* on L^2 is defined as $(x,y) \leq_p (u,v)$ if $x \leq u$ and $v \leq y$. If (u,v) is consistent, the latter means that (x,y) approximates all elements approximated by (u,v). If L is a complete lattice, then so is $\langle L^2, \leq_p \rangle$.

AFT studies fixpoints of lattice operators $O : L \to L$ through operators approximating O.

Definition 2.6.1. An operator $A : L^2 \to L^2$ is an *approximator* of O if

- A is $\leq_p$-monotone
- $\forall x \in L$: $O(x) \in [x', y']$, where $(x', y') = A(x,x)$.

Approximators map L^c into L^c. As usual, we restrict our attention to *symmetric* approximators: approximators A such that for all x and y, $A(x,y)_1 = A(y,x)_2$. Denecker et al. [2004] showed that the consistent fixpoints of interest (supported, stable, well-founded) are uniquely determined by an approximator's restriction to L^c, hence, sometimes we only define approximators on L^c. Given an approximator A, we can also derive the stable operator $S_A : L \to L : S_A(y) = \text{lfp}(A(\cdot,y)_1)$, where $A(\cdot,y)_1$ denotes the operator $L \to L : x \mapsto A(x,y)_1$.

AFT studies fixpoints of O using fixpoints of A.

- The *A-Kripke-Kleene fixpoint* is the $\leq_p$-least fixpoint of A and approximates all fixpoints of O.
- A *partial A-stable fixpoint* is a pair (x,y) such that $x = S_A(y)$ and $y = S_A(x)$.

- An *A-stable fixpoint* of O is a fixpoint x of O such that (x, x) is a partial A-stable fixpoint.
- The *A-well-founded fixpoint* is the least precise partial A-stable fixpoint.

2.6.3 Semantics for AEL using AFT

Denecker et al. [1998] showed that many semantics from AEL can be obtained by direct applications of AFT. In order to do this, they defined a three-valued version of semantic operator D_T.

In order to approximate an agent's state of mind, i.e., to represent partial information about possible world structures, Denecker et al. defined a belief pair as a tuple (P, S) of two possible world structures. They say that a belief pair *approximates* a possible world structure Q if $P \leq_K Q \leq_K S$, or equivalently if $S \subseteq Q \subseteq P$. Intuitively, P is an underestimation and S is an overestimation of Q. That is, P contains all interpretations that are potentially contained in the possible world structure, and S all interpretations that are certainly contained in the possible world structure. Stated even differently, P contains all knowledge the agent certainly has and S all knowledge the agent possibly has. We call a belief pair (P, S) *consistent* if $P \leq_K S$, i.e., if it approximates at least one possible world structure. From now on, we assume all belief pairs to be consistent. Belief pairs are ordered by a precision ordering $\leq_p$. Given two belief pairs (P, S) and (P', S'), we say that (P, S) is less precise than (P', S') (notation $(P, S) \leq_p (P', S')$) if $P \leq_K P'$ and $S' \leq_K S$.

We now define a three-valued valuation of sentences with respect to a belief pair (which represents an approximation of the state of mind of an agent) and a structure, representing the state of the world.

Definition 2.6.2. The *value* of φ with respect to belief pair B and interpretation I (notation $\varphi^{B,I}$) is defined inductively:

$$(P(\bar{t}))^{B,I} = \bar{t}^I \in P^I$$

$$(\neg\varphi)^{B,I} = \begin{cases} \mathbf{t} & \text{if } \varphi^{B,I} = \mathbf{f} \\ \mathbf{f} & \text{if } \varphi^{B,I} = \mathbf{t} \\ \mathbf{u} & \text{otherwise} \end{cases}$$

$$(\varphi \wedge \psi)^{B,I} = \begin{cases} \mathbf{t} & \text{if } \varphi^{B,I} = \mathbf{t} \text{ and } \psi^{B,I} = \mathbf{t} \\ \mathbf{f} & \text{if } \varphi^{B,I} = \mathbf{f} \text{ or } \psi^{B,I} = \mathbf{f} \\ \mathbf{u} & \text{otherwise} \end{cases}$$

$$(\forall x : \varphi)^{B,I} = \begin{cases} \mathbf{t} & \text{if } \varphi[x/d]^{B,I} = \mathbf{t} \text{ for all } x \in D \\ \mathbf{f} & \text{if } \varphi[x/d]^{B,I} = \mathbf{f} \text{ for some } x \in D \\ \mathbf{u} & \text{otherwise} \end{cases}$$

$$(K\varphi)^{(P,S),I} = \begin{cases} \mathbf{t} & \text{if } \varphi^{(P,S),I'} = \mathbf{t} \text{ for all } I' \in P \\ \mathbf{f} & \text{if } \varphi^{(P,S),I'} = \mathbf{f} \text{ for some } I' \in S \\ \mathbf{u} & \text{otherwise} \end{cases}$$

Example 2.6.3. We show how the three-valued valuation works, using the PWS's from Example 2.5.3. We define a belief pair $B = (Q', Q)$. Using this belief pair, we find that $K(\mathit{color} = \mathit{green})^B$ is false since there is a PWS (S_1) in Q where $\mathit{color} = \mathit{green}$ is not true. $(\mathit{shape} = \mathit{round})^B$ is true since $(\mathit{shape} = \mathit{round})^S = \mathbf{t}$ for all $S \in Q'$. Lastly, $K(\mathit{color} = \mathit{blue})^B$ is unknown, since there is no $S \in Q$ such that $(\mathit{color} = \mathit{blue})^S = \mathbf{f}$, but it is also not the case that $(\mathit{color} = \mathit{blue})^{S'} = \mathbf{t}$ for all $S' \in Q'$.

The logical connectives combine truth values based on Kleene's truth tables [Kleene, 1938]. Denecker et al. [2000] defined the bilattice operator D_T^* that maps (P, S) to (P', S') where

$$P' = \{I \mid T^{(P,S),I} \neq \mathbf{f}\} \text{ and } S' = \{I \mid T^{(P,S),I} = \mathbf{t}\}$$

P' contains all knowledge that can *certainly* be derived from the current state of mind and S' all knowledge that can *possibly* be derived from it. Denecker et al. showed that D_T^* is an approximator of D_T. The operators induce a class of semantics for AEL:

- Moore's expansion semantics (supported fixpoints).
- Kripke-Kleene expansion semantics [Denecker et al., 1998] (Kripke-Kleene fixpoints).

- (Partial) stable extension semantics [Denecker et al., 2003] ((partial) stable fixpoints).
- Well-founded extension semantics [Denecker et al., 2003] (well-founded fixpoints).

The latter two were new semantics induced by AFT.

3 Interactive Configuration

In the past decades enormous progress in many different areas of computational logic was obtained. This resulted in a complex landscape with many declarative paradigms, languages and communities. One issue that fragments computational logic more than anything else is the reasoning/inference task. Computational logic is divided in different declarative paradigms, each with its own syntactical style, terminology and conceptuology, and designated form of inference (e.g, deductive logic, logic programming, abductive logic programming, databases (query inference), answer set programming (answer set generation), constraint programming, etc.). Yet, in all of them declarative propositions need to be expressed. Take, e.g., “each lecture takes place at some time slot”. This proposition could be an expression to be deduced from a formal specification if the task was a verification problem, or to be queried in a database, or it could be a constraint for a scheduling problem. It is, in the first place, just a piece of information and we see no reason why depending on the task to be solved, it should be expressed in a different formalism (classical logic, SQL, ASP, MiniZinc, etc.).

This chapter presents a study with the KB system we introduced in the previous chapter. Here, we investigate the application of knowledge representation and reasoning to the problem of Interactive Configuration (IC) and analyse how the KB-paradigm behaves in a real-life context.

An interactive configuration problem [McDermott, 1982, Mittal and Frayman, 1989, Fleischanderl et al., 1998, Junker and Mailharro, 2003, Hadzic, 2004] is an interactive version of a constraint solving problem. One or more users search for a configuration of objects and relations between them that satisfies a set of constraints. Industry abounds with interactive configuration problems: configuring composite physical systems such as cars and computers, insurances, loans, schedules involving human interaction, webshops (where clients choose composite objects), etc. However, building such software is renowned in industry as difficult and no broadly accepted solution methods are available [Felfernig et al., 2014, Axling and Haridi, 1996]. Building software support using standard imperative programming is often a nightmare [Barker and O'Connor, 1989, Piller et al., 2014], due to the fact that (1) many functionalities need to be provided, (2) they are complex to implement, and (3) constraints on the configuration tend to get duplicated and spread out over the application, in the form of snippets of code performing various computations relative to the constraint (e.g., context dependent checks or propagations) which often leads to an unacceptable maintenance cost. This makes interactive configuration an excellent domain to illustrate the advantages of declarative methods over standard imperative or object-oriented programming.

The research question in this chapter is: how can we express the constraints of correct configurations in a declarative logic and provide the required functionalities by applying inference on this domain knowledge? This is a KRR question albeit a difficult one. In the first place, some of the domain knowledge may be complex. For an example in the context of a computer configuration problem, take the following constraint: *the total memory usage of different software processes that needs to be in main memory simultaneously, may not exceed the available RAM memory.* It takes an expressive knowledge representation language with aggregates to (compactly and naturally) express such a constraint. Many interactive configuration problems include complex constraints: various sorts of quantification, aggregates, definitions (sometimes inductive), etc. Moreover, an interactive configuration system needs to provide many functionalities: checking the validity of a fully specified configuration, correct and safe reasoning on a partially specified configuration (this involves reasoning on incomplete knowledge, sometimes with infinite or unknown domains), computing impossible values or forced values for attributes, generating sensible questions to the user, providing explanation why certain values are impossible, backtracking if the user regrets some choices, supporting the user by filling in his don't-cares while potentially taking into account a cost function, etc. That declarative methods are particularly suitable for solving this type of problem has been acknowledged before, and several systems and languages have been developed [Hadzic, 2004, Schneeweiss and Hofstedt, 2011, Tiihonen et al., 2013, Vlaeminck et al., 2009].

The contributions of this research are two-fold. First, we analysed IC problems from a knowledge representation point of view. We show that multiple functionalities in this domain can be achieved by applying different forms of logical inference on *the same* formal specification of the configuration domain. We define various sorts of inference and analyse them in terms of which different functionalities can be supplied. The second contribution is the reverse: we study the feasibility and usefulness of the KB paradigm in this important class of applications.

This chapter is structured as follows: first, we analyse the functional requirements for a good IC system, by identifying all required subtasks a good system should support. We then suggest a formalisation of these subtasks in the form of logical inferences on a central knowledge base. Using this, we build a proof of concept around a knowledge base to support a large configuration problem for a banking company. The logic used in this experiment is the logic FO(·) [Denecker and Ternovska, 2008], an extension of first-order logic (FO), and the system is the IDP system [De Cat et al., 2016]. We discuss the complexity of (the decision problems of) the inference problems and why they are solvable, despite the high expressivity of the language and the complexity of inference. We evaluated our approach using the evaluation criteria of the knowledge-based configuration research [Felfernig et al., 2014]. This chapter is then concluded with a discussion of related work in using knowledge-based systems for configuration and a comparison of our approach with these systems.

The work presented in this chapter was published as a conference paper at the Practical Aspects of Declarative Languages conference in [Van Hertum et al., 2016b] and later an extended version was published as a journal paper in Theory and Practice of Logic Programming [Van Hertum et al., 2016c].

3.1 Interactive Configuration

In an IC problem, one or more users search for a configuration of objects and relations between them that satisfies a set of constraints. Typically, the user is not aware of all constraints. There may be too many of them to keep track of. Even if the human user can oversee all constraints that he needs to satisfy, he is not a perfect reasoner and cannot comprehend all consequences of his choices. This in its own right makes such problems hard to solve. The problems get worse if the user does not know about the relevant objects and relations or the constraints on them, or if the class of involved objects and relations is large, if the constraints get more complex and more “irregular" (e.g., exceptions), if more users are involved, etc. On top of that, the underlying constraints in such

PriceOf		**PreReq**	
software	*int*	*software*	*software*
Windows	60	Office	Windows
Linux	20	LaTeX	Linux
LaTeX	10		
Office	30		
DualBoot	40		

MaxCost		**IsOS**
employee	*int*	*software*
Secretary	100	Windows
Manager	150	Linux

Table 3.1: Example data

problems tend to evolve quickly. All these complexities occur frequently, making the problem difficult for a human user. In such cases, computer assistance is needed: the human user chooses and the system assists by guiding him through the search space.

For a given IC problem, an IC system has information on that problem. There are a number of stringent rules to which a configuration should conform, and besides this there is a set of parameters. Parameters are the open fields in the configuration that need to be filled in by the user or decided by the system.

To illustrate all concepts used and introduced in this section, we will use a running example. This example is a simplified version of the application that was studied, that will be described in Section 3.3.1. We will use this example multiple times in Section 3.2.

We introduce the domain knowledge of this example here.

Example 3.1.1. Software on a computer has to be configured for different employees. Table 3.1 contains the information on the software, the requirements, the budgets of the employees and the prices of software. Available software is Windows, Linux, LaTeX, Office and a DualBoot system. Each software item has a price, which can be seen in column **PriceOf**. Column **PreReq** specifies which software is required for other software. Every type of employee has a budget, provided in column **MaxCost**. **IsOs** lists the pieces of software that are operating systems. Next to the information in the table, we know that if more than one OS is installed, a DualBoot System is required.

3.1.1 Subtasks of an interactive configuration system

Any system assisting a user in interactive configuration must be able to perform a set of subtasks. We look at important subtasks that an interactive configuration system should support.

Subtask 1: Acquiring information from the user

The first task of an IC system is acquiring information from the user. The system needs to get a value for a number of parameters of the configuration from the user. There are several options: the system can ask questions to the user, it can make the user fill in a form containing open text fields, dropdown-menus, checkboxes, etc. Desirable aspects would be to give the user the possibility to choose the order in which he gives values for parameters and to omit filling in certain parameters (because he does not know or does not care). For example, in the running example a user might need a LaTeX-package, but he does not care about which OS he uses. In that case the system will decide in his place that a Linux system is required. Since a user is not fully aware of all constraints, it is possible that he inputs conflicting information. This needs to be handled or avoided.

Subtask 2: Generating consistent values for a parameter

After a parameter is selected (by the user or the system) for which a value is needed, the system can assist the user in choosing these values. A possibility is that the system presents the user with a list of all possible values, given the values for other parameters and the constraints of the configuration problem. Limiting the user with this list makes that the user is unable to input inconsistent information.

Subtask 3: Propagation of information

Assisting the user in choosing values for the parameters, a system can use the constraints to propagate the information that the user has communicated. This can be used in several ways. A system can communicate propagations through a GUI, for example by coloring certain fields red or graying out certain checkboxes. Another way is to give a user the possibility to explicitly ask "what if"-questions to the system. In Example 3.1.1, a user can ask the system what the consequences are if he was a secretary choosing an Office installation. The system answers that in this case a Windows installation is required, which results in a Linux installation becoming impossible (due to budget constraints) and as a consequence it also derives the impossibility of installing LaTeX.

Subtask 4: Checking the consistency for a value

When it is not possible/desirable to provide a list of possible values, the system checks that the value the user has provided is consistent with the known data and the constraints.

Subtask 5: Checking a configuration

If a user makes manual changes to a configuration, the system provides him with the ability to check if his updated version of the configuration still conforms to all constraints.

Subtask 6: Autocompletion

If a user has finished communicating all his preferences, the system autocompletes the partial configuration to a full configuration. This can be done arbitrarily (a value for each parameter such that the constraints are satisfied) or the user can have some other parameters like total cost, that has to be optimized.

Subtask 7: Explanation

If a supplied value for a parameter is not consistent with other parameters, the system can explain this inconsistency to the user. This can be done by showing minimal sets of parameters with their values that are inconsistent, by showing (visualizations of) constraints that are violated or by combinations of both. It can also explain to the user why certain automatic choices are made, or why certain choices are impossible.

Subtask 8: Backtracking

It is not unthinkable that a user makes a mistake, or changes his mind after seeing consequences of choices he made. Backtracking is an important subtask for a configuration system, and can be supported in numerous ways. The simplest way is a simple back button, which reverts the last choice a user made.

A more involved option is a system where a user can select any parameter and erase his value for that parameter. The user can then decide this parameter at a later timepoint. Even more complex is a system where a user can supply a value for a parameter and if it is not consistent with other parameters the system shows him which parameters are in conflict and proposes other values for these parameters such that consistency can be maintained.

3.2 Interactive Configuration in the KB paradigm

To analyze the IC problem from the KB point of view, we aim at formalizing the subtasks of Section 3.1.1 as inferences. We will not deal with user interface aspects. For a given application, our knowledge base consists of a vocabulary Σ, a theory T expressing the configuration constraints and a partial structure $\mathcal{S}$. Initially, $\mathcal{S}_0$ is the partial structure that contains the domains of the types and the input data. During IC, $\mathcal{S}_0$ will evolve into more and more precise partial structures $\mathcal{S}_i$ due to choices made by the user. For IC, the KB also contains $L_{\mathcal{S}_0}$, the set of all uninterpreted domain atoms/terms[1] in $\mathcal{S}_0$. These domain terms are the logical formalization of the parameters of the IC problem. Σ and T are fixed. As will be shown in this section, all subtasks can be formalized by (a combination of) inferences on this knowledge base consisting of $\Sigma, T, \mathcal{S}_0, L_{\mathcal{S}_0}$ and information gathered from the user.

Example 3.2.1. Continuing Example 3.1.1, use vocabulary Σ:

$$\begin{aligned}
&\Sigma = \{\\
&\quad \Sigma_T = \{software,\ employee,\ int\}\\
&\quad \Sigma_P = \{Install(software),\ IsOS(software),\\
&\qquad\qquad PreReq(software, software)\}\\
&\quad \Sigma_F = \{PriceOf(software) : int,\ MaxCost(employee) : int,\\
&\qquad\qquad Cost : int,\ Requester : employee\}\\
&\}
\end{aligned}$$

[1]In the rest of this chapter, a domain atom is treated as a term that evaluates to true or false.

The initial partial structure $\mathcal{S}_0$ consists of:

$$\begin{aligned}
&\mathcal{S}_0 = \{\\
&\quad D_{\mathcal{S}_0} = \{Secretary, Manager, Windows, Linux, \text{\LaTeX},\\
&\qquad\qquad Office, DualBoot\} \cup \mathbb{Z}\\
&\quad employee^{\mathcal{S}_0} = \{Secretary, Manager\}\\
&\quad software^{\mathcal{S}_0} = \{Windows, Linux, \text{\LaTeX}, Office, DualBoot\}\\
&\quad int^{\mathcal{S}_0} = \mathbb{Z}\\
&\quad IsOs^{\mathcal{S}_0} = \{Windows, Linux\}\\
&\quad PreReq^{\mathcal{S}_0} = \{(Office, Windows), (\text{\LaTeX}, Linux)\}\\
&\quad MaxCost^{\mathcal{S}_0} = \{Secretary \to 100, Manager \to 150\}\\
&\quad PriceOf^{\mathcal{S}_0} = \{Windows \to 60, Linux \to 20, \text{\LaTeX} \to 10,\\
&\qquad\qquad Office \to 30, DualBoot \to 40\}\\
&\}
\end{aligned}$$

which is a formal representation of Table 3.1. All symbols from Σ that are not specified above are assumed to be fully unknown in $\mathcal{S}_0$.
The set of parameters $L_{\mathcal{S}_0}$ is:

$$\begin{aligned}
&\{Requester, Install(Windows), Install(Linux),\\
&\ \ Install(Office), Install(\text{\LaTeX}), Install(DualBoot), Cost\}
\end{aligned}$$

The theory T consists of the following constraints:

$$\begin{aligned}
&\forall s1\, s2 :\ Install(s1) \land PreReq(s1, s2) \Rightarrow Install(s2).\\
&\quad \text{// The total cost is the sum of the prices of all installed software.}\\
&Cost = sum\{s|Install(s)|PriceOf(s)\}.\\
&Cost \leq MaxCost(Requester).\\
&\exists s :\ Install(s) \land IsOS(s).\\
&Install(Windows) \land Install(Linux) \Rightarrow Install(DualBoot).
\end{aligned}$$

Subtask 1: Acquiring information from the user

Key in IC is collecting information from the user on the parameters. During the run of the system, the set of parameters that are still open changes. In our KB system a derived inference (a combination of the inferences as introduced in Section 2.4.2) is used to calculate this set of parameters. Complexity results of derived inferences stem from basic results formulated by Mitchell and Ternovska [2005] and the observation that modelchecking is polynomial in the size of the domain.

Definition 3.2.2. Calculating uninterpreted terms.
GetOpenTerms($T, \mathcal{S}$) is the derived inference with input a theory T, a partial

structure $\mathcal{S} \geq_p \mathcal{S}_0$ and the set $L_{\mathcal{S}_0}$ of terms. Output is a set of terms such that for every term t in that set, there exist models I_1 and I_2 of T that extend $\mathcal{S}$ $(I_1, I_2 \geq_p \mathcal{S})$ for which $t^{I_1} \neq t^{I_2}$. Or formally:

$$\{l | l \in L_{\mathcal{S}_0} \wedge \{d | (l = d)^{\mathcal{S}'} = \mathbf{u}\} \neq \emptyset \wedge \mathcal{S}' = Propagate(T, \mathcal{S})\}$$

The complexity of deciding whether a given set of terms A is the set of uninterpreted terms is in $\mathbf{\Delta_2^P}$.

Note that in practice it is often not needed to find all uninterpreted terms at once. The system can generate these dynamically, giving one term or a subset of terms can also be very useful for the user. When the Δ_2^P complexity becomes an issue, this can be a good approximative way to use this inference.

An IC system can use this set of terms in a number of ways. It can use a metric to select a specific term, which it can pose as a direct question to the user. It can also present a whole list of these terms at once and let the user pick one to supply a value for. In Section 3.3.1, we discuss two different approaches we implemented for this project.

Example 3.2.3. In Example 3.2.1, the parameters and domains are already given. Assume that the user has chosen the value *Manager* for *Requester*, true for *Install(Windows)* and false for *Install(Linux)*. The system will return *GetOpenTerms*$(T, \mathcal{S}) = \{$*Install(Office), Install(DualBoot), Cost*$\}$.

Subtask 2: Generating consistent values for a parameter

A domain element d is a possible value for term t if there is a model $I \geq_p \mathcal{S}$ such that $(t = d)^I = \mathbf{t}$.

Definition 3.2.4. Calculating consistent values.
GetConsistentValues$(T, \mathcal{S}, t)$ is the derived inference with input a theory T, a partial structure $\mathcal{S}$ and a term $t \in GetOpenTerms(T, \mathcal{S})$. Output is the set

$$\{t^I |\ I \text{ is a model of } T \text{ extending } \mathcal{S}\}$$

The complexity of deciding that a set P is the set of consistent values for t is in $\mathbf{\Delta_2^P}$.

Example 3.2.5. The consistent values for *Requester* given T and the initial partial structure $\mathcal{S}_0$ from Example 3.2.1 is:

$$GetConsistentValues(T, \mathcal{S}, Requester) = \{Secretary, Manager\}$$

Consistent values for other terms are the integers for *Cost* and $\{$*true, false*$\}$ for the others.

Subtask 3: Propagation of information

It is informative for the user that he can see the consequences of assigning a particular value to a parameter.

Definition 3.2.6. Calculating Consequences.
PosConsequences($T, \mathcal{S}, t, a$**)** and **NegConsequences(**$T, \mathcal{S}, t, a$**)** are derived inferences with input a theory T, a partial structure $\mathcal{S}$, an uninterpreted term t $\in$ GetOpenTerms$(T, \mathcal{S})$ and a domain element $a \in$ GetConsistentValues$(T, \mathcal{S}, t)$. As output it has a set C^+, respectively C^- of tuples (q, b) of uninterpreted terms and domain elements. $(q, b) \in C^+$, respectively C^- means that the choice a for t entails that q will be forced, respectively prohibited to be b. Formally,

$$\begin{aligned}C^+ = \{(q,b) \mid (q=b)^{\mathcal{S}'} = \mathbf{t} \wedge (q=b)^{\mathcal{S}} = \mathbf{u} \\ \wedge\ \mathcal{S}' = \mathit{Propagate}(T, \mathcal{S} \cup \{t=a\}) \\ \wedge\ q \in \mathit{GetOpenTerms}(T, \mathcal{S}) \setminus \{t\}\ \} \\ C^- = \{(q,c) \mid (q=c)^{\mathcal{S}'} = \mathbf{f} \wedge (q=c)^{\mathcal{S}} = \mathbf{u} \\ \wedge\ \mathcal{S}' = \mathit{Propagate}(T, \mathcal{S} \cup \{t=a\}) \\ \wedge\ q \in \mathit{GetOpenTerms}(T, \mathcal{S}) \setminus \{t\}\ \}\end{aligned}$$

The complexity of deciding whether a set P is C^+ or C^- is in $\mathbf{\Delta_2^P}$.

Example 3.2.7. Say the user has chosen $\mathit{Requester} = \mathit{Secretary}$ and wants to know the consequences of making $\mathit{Install}(\mathit{Windows})$ true. The output in this case contains $(\mathit{Install}(\text{\LaTeX}), \mathbf{f})$ in $\mathit{PosConsequences}(T, \mathcal{S}, t, a)$ and $(\mathit{Install}(\text{\LaTeX}), \mathbf{t})$ in $\mathit{NegConsequences}(T, \mathcal{S}, t, a)$ since this combination is too expensive for a secretary. Note that there is not always such a correspondence between the positive and negative consequences. For example, when deriving a negative consequence for *Cost*, this does not necessarily imply a positive consequence.

Subtask 4: Checking the consistency for a value

A value d for a term t is consistent if there exists a model of T in which $t = d$ that extends the partial structure representing the current state.

Definition 3.2.8. Consistency Checking.
CheckConsistency$(T, \mathcal{S}, t, d)$ is the derived inference with input a theory T, a partial structure $\mathcal{S}$, an uninterpreted term t and a domain element d. Output is a boolean b that represents whether $\mathcal{S}$ extended with $t = d$ still satisfies T. Formally we return $\mathbf{t}$ if

$$(\mathcal{S} \cup \{t^{\mathcal{S}} = d\}) \models T$$

and $\mathbf{f}$ otherwise. Complexity of deciding if a value d is consistent for a term t is in **NP**.

Example 3.2.9. If a user has chosen $Install(Windows)$ and $Install(\text{\LaTeX})$ to be true, then $Manager$ will and $Secretary$ will not be a consistent answer for $Requester$.

Subtask 5: Checking a total configuration

Once the user has constructed a 2-valued structure S and makes manual changes to it, he may need to check if all constraints are still satisfied. A theory T is checked on a total structure S by calling $Modelcheck(T, S)$, with complexity in **P**.

Subtask 6: Autocompletion

If a user is ready communicating his preferences (Subtask 1) and there are undecided terms left which he does not know or care about, the user may want to get a full configuration (i.e. a total structure). This is computed by modelexpand. In particular:

$$I = Modelexpand(T, \mathcal{S})$$

In many of those situations the user wants to have a total structure with, for example, a minimal cost (given some term representing the cost t). This is computed by minimize:

$$I = Minimize(T, \mathcal{S}, t)$$

Example 3.2.10. Assume the user is a secretary and all he knows is that he needs Office. He chooses $Secretary$ for $Requester$ and true for $Install(Office)$ and calls autocompletion. A possible output is a structure S where for the remaining parameters, a choice is made that satisfies all constraints, e.g., $Install(Windows)^S = \mathbf{t}$, $Install(DualBoot)^S = \mathbf{t}$ and the other $Install$ atoms false. This is not a cheapest solution (lowest cost). By calling minimize using cost-term $Cost$, the DualBoot is dropped.

Subtask 7: Explanation

A whole variety of options can be developed to provide different kinds of explanations to a user. If a user supplies an inconsistent value for a parameter, options can range from calculating an inconsistent subset of the theory T (1) to giving a proof of inconsistency as in [Pontelli and Son, 2006] (2), to calculating a partial subconfiguration that has this inconsistency (3). UnsatSubstructure is a logical inference for option 3.

Definition 3.2.11. Calculating inconsistent structures.
UnsatSubstructure($T, \mathcal{S}$) is a derived inference with input a theory T and a partial structure $\mathcal{S}$ that cannot be extended to a model of T and as output all (partial) structures $\mathcal{S}' \leq_p \mathcal{S}$ such that $\mathcal{S}'$ cannot be extended to a model I of T. Formally, we return:

$$\{\mathcal{S}' | \mathcal{S}' \leq_p \mathcal{S} \wedge \neg(\exists I \geq_p \mathcal{S}' \wedge I \models T)\}$$

Complexity of deciding if a set is an inconsistent substructure is in $\mathbf{co-NP}$.

Example 3.2.12. When a user has selected *Secretary* for *Requester* and has chosen $Install(Windows)$ and $Install(\text{\LaTeX})$ to be true, this results in a partial structure $\mathcal{S}_1$ that cannot be extended to a model of theory T, in this case, for example $\mathcal{S}_1$ can be an element of **UnsatSubstructure($T, \mathcal{S}_1$)**:

$$\begin{aligned}
&\mathcal{S}_1' = \{\\
&\quad D_{\mathcal{S}_0} = \{Secretary, Manager, Windows, Linux, \text{\LaTeX},\\
&\qquad\qquad Office, DualBoot\} \cup \mathbb{Z}\\
&\quad employee^{\mathcal{S}_0} = \{Secretary, Manager\}\\
&\quad software^{\mathcal{S}_0} = \{Windows, Linux, \text{\LaTeX}, Office, DualBoot\}\\
&\quad int^{\mathcal{S}_0} = \mathbb{Z}\\
&\quad IsOs^{\mathcal{S}_0} = \{Windows, Linux\}\\
&\quad PreReq^{\mathcal{S}_0} = \{(Office, Windows), (\text{\LaTeX}, Linux)\}\\
&\quad MaxCost^{\mathcal{S}_0} = \{Secretary \rightarrow 100, Manager \rightarrow 150\}\\
&\quad PriceOf^{\mathcal{S}_0} = \{Windows \rightarrow 60, Linux \rightarrow 20, \text{\LaTeX} \rightarrow 10,\\
&\qquad\qquad Office \rightarrow 30, DualBoot \rightarrow 40\}\\
&\quad software_t^{\mathcal{S}_0} = \{Windows\}\\
&\}
\end{aligned}$$

The inference in Definition 3.2.13 calculates an inconsistent subtheory.

Definition 3.2.13. Calculating inconsistent theories.
UnsatSubtheory($T, \mathcal{S}$) is a derived inference with input theory T and a partial structure $\mathcal{S}$ such that there does not exist a model I, extending S, satisfying T.

The inference has as output all theories T' such that $T' \subseteq T$ and there is no model satisfying T', extending $\mathcal{S}$. Formally, we return:

$$\{T' | T' \subseteq T \wedge \neg(\exists I \geq_p \mathcal{S} \wedge I \models T')\}$$

Complexity of deciding if a theory is such an inconsistent theory is in $\mathbf{co-NP}$.

Example 3.2.14. When a user has selected *Secretary* for *Requester* and has chosen $Install(Windows)$ and $Install(\text{\LaTeX})$ to be true, this results in a partial structure $\mathcal{S}_1$ that cannot be extended to a model of theory T, in this case, for example T' can be an element of **UnsatSubheory**$(T, \mathcal{S}_1)$:

// The total cost is the sum of the prices of all installed software.
$Cost = sum\{s|Install(s)|PriceOf(s)\}.$
$Cost \leq MaxCost(Requester).$
$\exists s: \; Install(s) \wedge IsOS(s).$
$Install(Windows) \wedge Install(Linux) \Rightarrow Install(DualBoot).$

Note that Definition 3.2.11 and 3.2.13 do not make any statements of minimality.

Using the associated theory $T_\mathcal{S}$ (Section 2.2) and domain structure $\mathcal{S}_D$ of a partial structure $\mathcal{S}$, it is possible to consider calculating minimally precise partial configurations as a special case of calculating a minimal inconsistent subset of the theory. As in [Shchekotykhin et al., 2014], we can introduce a "background theory" $B \subset T \cup T_\mathcal{S}$ (a subset of the theory in which there are assumed to be no conflicts). We define multiple derived logical inferences, with different degrees of minimality (not-minimal, subset-minimal and minimal in size) of increasing complexity, able to provide explanations to the user.

Definition 3.2.15. Calculating inconsistent theories with a background.
UnsatSubtheory$(T, \mathcal{S}, B)$ is a derived inference with input theory T, a partial structure $\mathcal{S}$ and a background theory $B \subseteq T \cup T_\mathcal{S}$ such that there does not exist a model I, with the domains as in $\mathcal{S}_D$ satisfying $T \cup T_\mathcal{S}$ (or equivalently: extending $\mathcal{S}$ and satisfying T), but there is a model satisfying B. The inference has as output all theories T' such that $B \subseteq T' \subseteq T \cup T_\mathcal{S}$ and there is no model satisfying T'. Formally, we return:

$$\{T' | B \subseteq T' \subseteq (T \cup T_S) \wedge \neg(\exists I \geq_p \mathcal{S}_D \wedge I \models T')\}$$

Complexity of deciding if a theory is such an inconsistent theory is in $\mathbf{co-NP}$.

Definition 3.2.16. Calculating minimal inconsistent theories with a background.
MinimalUnsatTheory$(T, \mathcal{S}, B)$ is a derived inference with input theory T,

a partial structure $\mathcal{S}$ and a background theory B as above. Output is the subset of subset minimal theories from *UnsatSubtheory*$(T, \mathcal{S}, B)$. Complexity of deciding if a set is a subset minimal inconsistent theory is in Δ_2^P.

Definition 3.2.17. Calculating minimum inconsistent theories with a background.
MinimumUnsatTheory$(T, \mathcal{S}, B)$ is a derived inference with input theory T, a partial structure $\mathcal{S}$ and a background theory B as above. Output is the subset of cardinality minimal theories from *MinimalUnsatTheory*$(T, \mathcal{S}, B)$. Complexity of deciding if a set is a cardinality minimal inconsistent theory is Π_2^P.

Note that Definition 3.2.11 is equivalent to calculating a minimal inconsistent subset of a theory $T \cup T_{\mathcal{S}}$, with $B = T$, if you translate the output back to a pair of a theory and a structure. Definition 3.2.13 is equivalent to calculating a minimal inconsistent subset of a theory $T \cup T_{\mathcal{S}}$, with $B = T_{\mathcal{S}}$, if you translate the output back to a pair of a theory and a structure.

In literature multiple approaches are discussed, all mapping to one of our explanation-related inferences. QuickXPlain [Junker, 2004] is an algorithm that calculates elements of Definition 3.2.15. The Hitting Set Directed Acyclic Graph (HSDAG) [Reiter, 1987] algorithm calculates subset minimal inconsistent theories (Definition 3.2.16), as in different ASP solvers [Shlyakhter et al., 2003, Syrjänen, 2006]. Implementations of Definition 3.2.17 have been described in [Lynce and Silva, 2004] and [Zhang et al., 2006]. In our experiment, we have an implementation of Definition 3.2.16 [Wittocx et al., 2009], where we, however, do not calculate the entire set of subset minimal theories. We only calculate one, which gives one explanation of the inconsistency. If the user resolves that problem, he can ask for a new explanation which will point to another reason of inconsistency. This process is reiterated until all problems are resolved.

Example 3.2.18. We show a minimal inconsistent subtheory in a situation with T as in Example 3.2.1 and $\mathcal{S}_i$, a partial structure representing an intermediate configuration where a user started with $\mathcal{S}_0$ and has chosen *Secretary* for *Requester*, and wants to Install *Office* and *Linux*. This is not possible, and as such, the user asks the system for an explanation in the form of a minimal inconsistent theory. A possible minimal inconsistent theory with $B = \emptyset$, is:

$$Install(\mathit{Office}).$$

$$(Install(\mathit{Office}) \wedge \mathit{PreReq}(\mathit{Office}, Windows)) \Rightarrow Install(Windows).$$

$$\mathit{Cost} = \mathit{sum}\{(s, \mathit{PriceOf}(s)) | Install(s)\}.$$

$$\mathit{Cost} \leq MaxCost(\mathit{Requester}).$$

This means that there is no valid configuration because Windows needs to be installed as prerequisite for Office, and the total cost then exceeds the budget of a Secretary.

Subtask 8: Backtracking

If a value for a parameter is not consistent, the user has to choose a new value for this parameter, or backtrack to revise a value for another parameter. In Section 3.1.1 we discussed three options of increasing complexity for implementing backtracking functionality. Erasing a value for a parameter is easy to provide in our KB system, and since this is a generalization of a back button (erasing the last value) we have a formalization of the first two options. Erasing a value d for parameter t in a partial structure $\mathcal{S}$ is simply modifying $\mathcal{S}$ such that $(t = d)^{\mathcal{S}} = \mathbf{u}$. As with explanation, a number of more complex options can be developed. We look at one possibility. Given a partial configuration $\mathcal{S}$, a parameter p and a value d that is inconsistent for that parameter, calculate a minimal set of previous choices that need to be undone such that this value is possible for this parameter. The converse of this problem is well known under the name of maximum satisfiability problems. In other words, you want to hold on to as much of the structure as possible while ensuring satisfiability.

This problem is closely related to the explanation subtask [Heras et al., 2011, Marques-Silva and Planes, 2008]. You can imagine the explanation problem as asking the system to point out a mistake in your reasoning. However, solving this mistake will not guarantee you have not made any other mistake in the rest of the problem. What we actually need is a minimal set of things we can remove, so every problem is solved simultaneously.

So more formally, we can use Definition 3.2.11 and calculate $UnsatStructure(T \wedge (t = d), \mathcal{S})$. This inference calculates a set A of sets of previous choices that together are inconsistent. Undoing an arbitrary choice in all of these sets results in a partial subconfiguration $\mathcal{S}'$ of $\mathcal{S}$ such that d is a possible value for t in $\mathcal{S}'$. To find the maximal partial subconfiguration $\mathcal{S}'$ that satisfies that property, the minimal hitting set [Reiter, 1987] of all sets in A has to be calculated.

3.3 Proof of Concept

3.3.1 Implementation

In this section we will describe the developed application and implementation. Our application has a simple client-server architecture. The server plays the role of the reasoning engine, which is mainly a thin wrapper around the IDP system. The client consists of a GUI made in QML [QML] as front-end.

The server converts IDP into a stateless server which is accessible through the web. The client application sends the necessary information, consisting of theories, partial structures and choices, to this server and the server executes the needed inferences. This is a design which involves repeatedly sending over the choices a user has made, but it allows for a very simple architecture to show the feasibility of our design.

This implementation was done in cooperation with Adaptive Planet, a consulting company [Adaptive Planet] that developed the user interface, and an international banking company that provided us with a substantial configuration problem for testing purposes. More practical information about this implementation, some screenshots, a downloadable demo and another example of a configuration system developed with IDP as a reasoning engine (a simpler course configuration demo) can be found at: `http://www.configuration.tk`.

The Reasoning Engine

As explained before, the application we developed was built on the knowledge base system IDP, which was not developed specifically with configuration problems in mind. It provides the basic inferences listed at the end of Section 2.4.2. The goal of this experiment was to check if this general infrastructure could be readily applied to applications such as configuration.

In Section 3.2 we showed how the tasks which are needed for configuration relate to the infrastructure provided by IDP. Our main implementation task was to convert these specifications to code. Some subtasks such as autocompletion did not require any extra work, as this functionality is directly available as the modelexpand inference. Some functionality, e.g. calculating consequences, did require some work but the existing functionality provided almost all needed components.

We mainly use the existing forms of inference that are readily available in the IDP system. No dedicated or specialized algorithms are used for the configuration subtasks. This proves the point that the KB-paradigm is very flexible but this also means that we had relatively little impact upon the efficiency of our server. However, the system ended up being quite responsive and we could conclude that IDP (and by extension the KB-paradigm) passed the test for usefulness in this application.

User Interface

Apart from a reasoning engine, it is also necessary to have an accessible front end so the user has easy access to the multitude of functionalities which are available. The front end consists of an application written in the Qt framework using QML [QML] and connects to a configuration engine over the web. For the purposes of our demo, we developed two different graphical interfaces:

Wizard In the wizard interface, the user is interrogated and he answers on subsequent questions selected by the system, using the $GetOpenTerms$ inference. An important side note here is that the user can choose not to answer a specific question, for instance because he cannot decide as he is missing relevant information or because he is not interested in the actual value (at this point). These parameters can be filled in at a later timepoint by the user, or by the system, using propagation, or in case the user calls autocompletion.

Drill-Down In the drill-down interface, the user sees a list of the still open parameters, and can pick which one he wants to fill in next. This interface is useful if the user is a bit more knowledgeable about the specific configuration and wants to give the values in a specific order.

In both interfaces the user is assisted in the same way when he enters data. When he or the system selects a parameter, he is provided with a dropdown list of the possible values, using the $GetConsistentValues$ inference. Before committing to a choice, he is presented with the consequences of his choice, using the calculate consequences inference. The nature of the system guarantees a correct configuration and will automatically give the user support using all information it has (from the knowledge base, or received from the user).

3.3.2 Evaluation

Evaluation Criteria

When evaluating the quality of software (especially when evaluating declarative methods), scalability (data complexity) is often seen as the most important quality metric. Naturally when using an interactive configuration system, performance is important. However, in the configuration community it is known that reasoning about typical configuration problems is relatively easy and does not exhibit real exponential behavior [Tiihonen et al., 2013]. Also, depending on the application, it is reasonable to expect the number of parameters to be limited, since humans need to fill in the configuration in the end. When developing a configuration system, challenges lie in the complexity of the knowledge, its high volatility and the complex functionalities to be built. To get a more complete view of the performance of a configuration system, we chose to evaluate on a larger set of different evaluation criteria. In recent literature [Felfernig et al., 2014] nine evaluation criteria are used to differentiate between different paradigms used for configuration. In Section 3.4, ten other existing approaches will be discussed and compared to our solution using the same nine criteria.

Grapical Modeling Concepts (C1) is supported if there are standard graphical modeling techniques available that visualize configuration knowledge. They improve understandability, development time and maintenance of new knowledge bases.

Component Oriented Modeling (C2) is a criterion that states that the modeling language is a natural language that allows knowledge base design on the basis of real-world concepts: types, relations, hierarchies, etc.

Automated Consistency Maintenance (C3) can be broken down to two categories. Firstly, a system can have support for a priori automated consistency maintenance. This helps a developer write consistent constraints and verifying correctness while writing the knowledge base. Secondly, runtime automated consistency maintenance supports the end user, by guaranteeing that every intermediate configuration he can make, can be extended to a valid configuration.

Modularization Concepts are available (C4) if the modeling language is modular and has support for adding additional structure to the knowledge base, for example by organizing the constraints in blocks or groups.

Maintainability (C5) relates to the adaptability of the knowledge base if the background information changes. The background information is volatile, it is for example depending on ever-changing company policies. As such, it is vital that when that information changes, the system can be easily adapted. When using custom software, all tasks using domain knowledge (like rules and policies) need their own program code. The domain knowledge is scattered all over the program. If this policy changes, a programmer has to find all snippets of program code that are relevant for guarding this policy and modify them. This results in a system that is hard to maintain, hard to adapt and error-prone. Every time the domain knowledge changes, a whole development cycle has to be run through again. Some systems have support for intelligent knowledge base navigation tools for complex knowledge spaces.

Model-based (C6) means that a knowledge base in the system expresses exactly what it means for a configuration to be valid. This in contrast to rule-based configuration, where a knowledge base also contains problem solving knowledge (i.e. information on how the rules should be used/fired).

Efficiency (C7) relates to efficiency and scalability of the reasoning engine.

Ability to solve generative problem settings (C8) means that the language supports talking about component types instead of specific objects. A system supports generic constraints if it allows for constraints that apply to every instance of a component type on which the constraint is defined. For example, the first constraint of Theory T in Example 3.2.1 is a generic constraint about all software, without explicitly naming the individual pieces of software.

Ability to provide explanations (C9) means that the system is able to communicate reasons for inconsistencies or explain why certain choices are forced/prohibited.

Evaluation

The criteria discussed in previous section are a good way to evaluate the KB implementation of a configuration system. We evaluate our implementation and the IDP system with these criteria.

Grapical Modeling Concepts (C1). IDP has no support for graphical modeling of domain knowledge and we did not develop any tools for this experiment. However, it must be noted, that a highly expressive and readable modeling language often makes graphical modeling obsolete.

Component Oriented modeling (C2). The FO(·) language used in this experiment is an extension of typed first-order logic. First-order logic uses small set of connectives: $\land, \lor, \neg, \Rightarrow, \Leftrightarrow, \exists, \forall$. These connectives are also the basic connectives of information used by humans. Classical logic is a good KR language because it has a very clear informal semantics. As is well known, there are major shortcoming for it to be used for knowledge representation. FO(·) extends classical logic with a number of extensions that arise from research in AI and KR, such as aggregates, inductive definitions, types, ... This makes FO(·) a suited modeling language for a configuration system.

Automated Consistency Maintenance (C3). A priori consistency maintenance is supported in the implementation by using the explanation inferences. If the developer has a collection of constraints that is consistent, it is possible to evaluate if a new constraint leads to an inconsistency and ask the system what other constraints it conflicts with, using for example definition 3.2.16. At runtime consistency maintenance is partially supported, by using the inferences in subtask 2, 3 and 4. These inferences are theoretically able to guarantee consistency, but due to computational limitations, approximate versions can be used. These are not always able to give the same guarantees.

Modularization concepts are available (C4). The implemented configuration system is modular, since a knowledge base can consist of multiple theories and structures, that together make up the specification. The explanation inference allows that a user selects background constraints, as in definition 3.2.16, and in this way he can choose about which constraints he needs feedback.

Maintainability (C5). The development of a KB system with a centrally maintained knowledge base makes the knowledge directly available, readable and adaptable. A well-known advantage of this approach is in maintainability: if domain information changes, the developer can easily modify the knowledge base. The current implementation does however have no additional support for knowledge base navigation tools.

Model-based (C6). The FO(·) modeling methodology is based on formulating the properties of a correct configuration in a natural way, such that the models of a specification correspond with configurations. This is inherently a model-based approach.

Efficiency (C7). As explained in Section 3.3.1, we have only written a thin layer upon existing software which did not target configuration problems specifically. The performance of the IDP system has been tested extensively in other contexts [Jansen et al., 2014, Bruynooghe et al., 2015]. The reasoning engine for IDP is very similar in performance to mainstream ASP solvers [Calimeri et al., 2014]. Their performance was tested more extensively in the context of configuration by Tiihonen et al. [2013]. It is also very difficult to reliably compare the response times for interactive systems. Standard benchmarking techniques in software engineering traditionally use instances which need multiple minutes to solve. In this setting we aim for subsecond response times, for which no standard benchmarks are available as far as we are aware.

In this experiment (a configuration task with 300 parameters and 650 constraints), our users reported a response time of a half second on average with outliers up to 2 seconds. Note that the provided implementation was a naive prototype and optimizing the efficiency of the implemented algorithms is still possible in a number of ways.

Ability to solve generative problem settings (C8). FO(·) is an extension of first-order logic, and as such has native support for quantification which is needed for generative problem settings.

Ability to provide explanations (C9). Subtask 7 and 8 in Section 3.2 are inferences that are used to support giving explanations. The implemented configuration system has an implementation of definition 3.2.16.

3.4 Related Work

3.4.1 Other approaches

In different branches of AI research, people have been focusing on configuration software in different settings. The following discussion of knowledge-based approaches is based on a book in recent literature [Felfernig et al., 2014]. After the discussion we will compare the ten approaches with our approach (**IDP**).

Historically, the first knowledge-based configuration systems were *rule-based* (**RBS**) [McDermott, 1982, Barker and O'Connor, 1989]. These systems operate on a working memory and if the condition of a rule is fulfilled, it fires and modifies the working memory, applying the conclusion of that rule. Rule-based systems are sensitive to rule orderings. This complicates modification of the rule-base. More importantly, inclusion of problem solving knowledge in the rule-base, makes a rule-base problem specific and focused towards one specific task. This leads to the same problems as in imperative languages. To solve different tasks, more rule-bases have to be built, leading to duplication and fanning out of knowledge, giving issues in maintainability.

Constraint Satisfaction Problems are widely used for tackling configuration problems [Mittal and Frayman, 1989, Fleischanderl et al., 1998]. A (static[2]) constraint satisfaction problem (**SCSP**) is a triple (V, D, C) of a set of domain variables $V = \{v_1, v_2, \ldots, v_n\}$, a set of domains $\{dom(v_1), dom(v_2), \ldots, dom(v_n)\}$ and set of constraints C. A solution for a SCSP is an assignment S of domain elements $d_i \in dom(v_i)$ to variables v_i, such that each variable has a value in S and constraints C are satisfied by S. A configuration task in SCSP is searching for a solution for a SCSP (V, D, C), where C contains the configuration constraints together with the user preferences. To make efficient CSP configuration systems, different techniques have been used, such as local search [Li et al., 2005], symmetry breaking [Kiziltan et al., 2001] and knowledge compilation techniques such as binary decision diagrams [Hadzic and Andersen, 2005]. In response to limitations of SCSP in configuration, extensions have been developed. *Dynamic Constraint Satisfaction Problems* (**DCSP**) [Mittal and Falkenhainer, 1990] allow for variables to be inactive or irrelevant. If a variable is inactive, it does not need a value in a solution (for example, when configuring a smartphone, no camera resolution is needed if no camera is present). *Generative Constraint Satisfaction Problems* (**GCSP**) [Fleischanderl et al., 1998] extends SCSP with component types and generative constraints.

Janota [2008] studied a mapping of CSP to SAT to use a SAT solver to provide functionality for a configuration system.

There exist many graphical approaches for doing knowledge configuration, and visualizing a configuration model. Kang [1990] used *feature models* (**FM**) for modeling these concepts, while **UML** was proposed in [Falkner and Haselböck, 2013]. FM and UML configuration approaches have no reasoning algorithms, they need to be used with external algorithms. Karatas et al. [2010] for example combined feature models with constraint logic programming (CLP) to provide reasoning and automated analysis.

Decidable subsets of first-order logic, *description logics* (**DL**) are used often in context of the semantic web. They have also been used for the development of configuration systems [Hotz et al., 2006, McGuinness and Wright, 1998]. The trade-off for having decidable subsets of first-order logic is that they are limited in expressivity. This make domain knowledge in these systems less readable, less natural and harder to maintain. An ontology based method was also proposed by Vanden Bossche et al. 2007 using OWL.

[2]In constrast to dynamic and generative constraint satisfaction problem.

Tiihonen et al. developed a configuration system WeCoTin [Tiihonen et al., 2013], based on *Answer Set Programming* (**ASP**). WeCoTin uses Smodels, an ASP system, as inference engine, for propagating consequences of choices. Answer set programming (ASP) is a form of declarative programming based on the stable-model semantics [Gelfond and Lifschitz, 1988] for logic programs. The architecture of their reasoning engine is closely related to the reasoning engine we use. Also, in language, many similarities can be identified [Denecker et al., 2012], as they both have their roots in extended logic programming.

Combinations of the above approaches are also proposed in literature, called *hybrid* (**HB**) configuration systems. Typically, they use a DL-based representation for the ontology, together with constraints. They combine reasoning engines from these fields to provide inference [Hotz et al., 2006].

3.4.2 Comparison of approaches

Felfernig et al. [2014] evaluated all these paradigms with respect to the evaluation criteria from Section 3.3.2. In Table 3.2, we show this evaluation, together with scores for our implementation in the **IDP** column, based on the discussion of Section 3.3.2.

Table 3.2: Comparison of systems from Section 3.4 using criteria from Section 3.3.2 as in (Felfernig et al. 2014). We use a ✓to mark good support, a $\approx$ for partial support and a $-$ to denote that no support is available.

C	RBS	SCSP	DCSP	GCSP	SAT	FM	UML	DL	ASP	HB	IDP
1	-	-	-	-	-	✓	✓	≈	-	≈	-
2	-	-	-	✓	-	-	✓	✓	✓	✓	✓
3	-	≈	≈	≈	≈	-	-	≈	≈	≈	≈
4	≈	-	-	✓	-	-	✓	✓	✓	✓	✓
5	-	≈	≈	≈	≈	≈	≈	≈	≈	≈	✓
6	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	-	-	≈	≈	≈	≈
8	≈	-	-	✓	-	-	-	-	≈	✓	✓
9	≈	✓	✓	✓	≈	-	-	✓	✓	✓	✓

All these approaches are focused towards one specific inference: ontologies are focused on deduction, rule systems are focused on backward/forward chaining, etc. These approaches are less general then the KB paradigm, which is specifically designed to reuse the knowledge for different reasoning tasks. The contributions of this work are different from previously discussed approaches:

we analyzed IC problems from a Knowledge Representation point of view. This chapter is a discussion of possible approaches and the importance of this point of view. We made a study of desired functionalities for an IC system and how we can define logical reasoning tasks to supply these functionalities. As far as we are aware, the language we used in this experiment is more expressive than earlier approaches.

The expressivity of the language is crucial for the usability of the approach. It allows us to address a broader range of applications, moreover it is easier to formalize and maintain the domain knowledge. Not discussed by Felfernig et al. [2014] et al is work by Vlaeminck et al. [2009]. They did a preliminary experiment using the KB approach for interactive configuration, also using the FO(·) IDP project. It is on this work that we continue in this work by analyzing a real-life application of a larger scale and discussing new functionalities and inferences. This theoretical approach benefits from (1) the expressive language to express domain knowledge adequately and (2) the general basic inferences that realise derived inferences in an easy way, supporting the discussed functionalities, resulting in a IC system that scores very well with relation to the evaluation criteria (Table 3.2).

An interesting remark in Table 3.2 is that the IDP column resembles the GCSP column, a generalisation of CSP, developed for configuration. The IDP-system has better support for C5 (maintainability), due to the high level modeling language and the strict seperation between domain knowledge and reasoning. GCSP has better efficiency results. This can be partly explained by the fact that CSP uses dedicated algorithms for reasoning over global constraints such as *alldifferent*. The goal of reusing knowledge makes that we typically do not make use of this kind of specific algorithms, since a dedicated algorithm can only be developed with one specific inference in mind.

3.5 Challenges and Future Work

Interactive configuration problems are part of a broader kind of problems, namely service provisioning problems. Service provisioning is the problem domain of coupling service providers with end users, starting from the request until the delivery of the service. Traditionally, such problems start with designing a configuration system that allows users to communicate their wishes, for which we provided a knowledge-based solution. Once all the information is gathered from a user, it is still necessary to make a plan for the production and delivery of the selected configuration. Hence the configuration problem is followed by a planning problem that shares domain knowledge with the configuration problem but that also has its own domain knowledge about providers of components, production processes, etc. This planning problem then leads to a monitoring problem. Authorizations could be required, payments need to be checked, or it could be that the configuration becomes invalid mid-process. In this case the configuration needs to be redone, but preferably without losing much of the work that is already done. Companies need software that can manage and monitor the whole chain, from initial configuration to final delivery and this without duplication of domain knowledge. This is a problem area where the KB approach holds great promise but where further research is needed to integrate the KB system with the environment that the company uses to follow up its processes.

Other future work may include language extensions to better support configuration-like tasks. A prime example of this are templates [Dasseville et al., 2015]. Oftentimes the theory of a configuration problem contains lots of constraints which are similar in structure. It seems natural to introduce a language construct to abstract away the common parts. Another useful language extension is reification, to talk about the symbols in a specification rather than about their interpretation. Reification allows the system to reason on a meta level about the symbol and for example assign symbols to a category like “Technical” or “Administrative”.

3.6 Conclusion

In this chapter, a large application was studied to analyse how the KB-paradigm behaves in a real-life context. We studied a class of knowledge intensive problems: interactive configuration problems. As we discussed why solutions for this class are hard to develop, we proposed a novel approach to the configuration problem based on an existing KB system.

First, we analysed the functional requirements for a good IC system. In total, 8 different subtasks were identified. These subtasks are all vital and together make up a well-working IC system. The goal of this research was to investigate how to develop a system that provides support for all these subtasks, by logical inferences on a knowledge base.

For this, we built a knowledge base to support a large configuration problem for a banking company, designed to configure newly bought servers. Using this, we identified interesting new inference methods and applied them to the interactive configuration domain. Using a set of base inferences, provided by an existing KB system, implementations were made for the new inferences. With these new implementations and the knowledge base, we built a proof of concept to solve the configuration problem for the banking company.

To evaluate, we studied our new proposed approach and compared it with 10 earlier approaches discussed in standard literature. Using an existing set of criteria, used for comparison between these 10 approaches, we evaluated our proposal. Some main advantages we found were the set of problems that could be solved, because of the richness of the language and the fact the KB approach is model based.

4 The KB Paradigm applied to Business Rules

As a second application and test-case for the IDP KBS system in this thesis, we look to the domain of Business Rules. Rule-based approaches are well-represented in industry for handling knowledge-intensive applications. We compared our KBS approach to a rule-based approach by focusing on a standard benchmark from the Business Rule domain: the EU-rent application [Agrawal, 2011]. We compared functionality by studying two different use cases in the application.

The first use case is the optimal planning and allocation of cars to a given collection of reservations. Because our system is well-suited to solving search problems of this kind, we expect that we can perform this task quite easily. We develop a new derived inference for this application which will allow us to revise models (or in this specific context, plannings) with penalties (or rewards) for certain modifications.

The other use case is the purchase of a new car (or the decision of the system to purchase a new car when certain conditions are satisfied). This second use case may seem like a simple operation, but it will actually turn out to be more difficult to model in the declarative language FO($\cdot$). This is because most of the inference tasks that are supported by our current KBS are concerned with generating or expanding models of a theory, given a fixed set of objects that is to serve as the domain of these models. However, in this use case, the set of cars can be modified, which will require us to extend the language. We introduce a *new* operator, stating when a new domain element should be created.

The work presented in this chapter was published as a technical communications paper at the International Conference on Logic Programming in [Van Hertum et al., 2013b] and as a technical report in [Van Hertum et al., 2013a]. A relevant abstract was published in in [Van Hertum, 2014]

4.1 Business Rules

Business Rules encode information in the form of sets of procedural style "IF-THEN" rules. The rules derive their (operational) meaning from the way they are interpreted in the match-resolve-act cycle, in which the THEN part of rules may be derived or executed given the satisfaction of the IF-part at some stage in the process.

Multiple reasoning engines for this formalism such as ILog JRules[JRules, 2005] and Jess[Friedman-Hill, 2003] have been developed. Due to the imbedded operational semantics in these rules, it is hard to reuse this knowledge for any other application than the chaining algorithm for which they were developed.

4.1.1 The Car Rental application

EU-Rent is a fictitious car rental company spread over different countries. It has a collection of cars that are located in specific branches. They are allowed to be moved between the branches and a customer can rent a car in one branch and return it to another branch. The application has to process all reservations and make a planning so that cars are at the right branch on the right time. Moreover, it should fulfill as many reservations as possible, while minimising the expenses. Next to this obvious task, it should be able to process smaller changes to the database, like adding or deleting a reservation, buying or selling a car, ...

The EU-Rent application is a well-known benchmark application used in literature to evaluate Business Rule systems. This application provides a simple but varied class of use cases to evaluate the expressivity and functionality of such systems.

Figure 4.2 represents the domain model of the modelled EU-Rent application. For readability, we use a smaller subset in this text, which can be seen in

Figure 4.1. This domain model consists of four types of objects: drivers, cars, reservations and branches. Each car has a purchase date and a mileage count. For each driver we know whether he or she has a driver's license. Reservations are requests to hire a car for a certain period, denoted by a start and end date. Reservations can be accepted or rejected, which is represented by a boolean attribute.

The objects themselves are related to each other, represented by the arrows in Figure 4.1. Every reservation has one or multiple drivers associated with it, can have a car allocated to it, that is requested to be picked up and dropped of at a certain branch of the company. Every car is stationed at a certain branch.

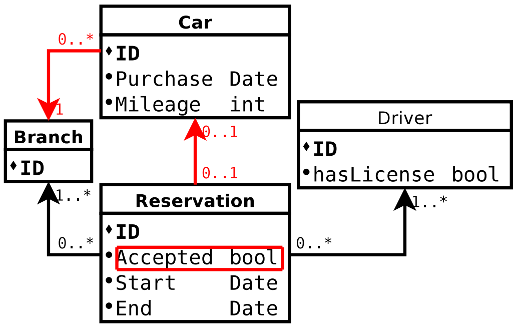

Figure 4.1: Restricted domain model of the modelled EU-Rent application

4.1.2 The domain knowledge

Static domain knowledge

A car rental system maintains a state recording available cars, reservations, planning of car transports, information about clients, etc. In this section, we describe the states and their invariants, using a vocabulary based on the domain specification. Figure 4.2 gives an overview of the domain knowledge, relevant for EU-Rent application. For readability, we will use a subset of that domain knowledge, represented in Figure 4.1.

Making a FO(·) specification of the relevant domain knowledge, starts with choosing a good vocabulary. Our modelling of the EU-Rent domain knowledge will use vocabulary SingleState (Figure 4.3) and maintain a structure over SingleState, representing the current state of the system. Even though we are

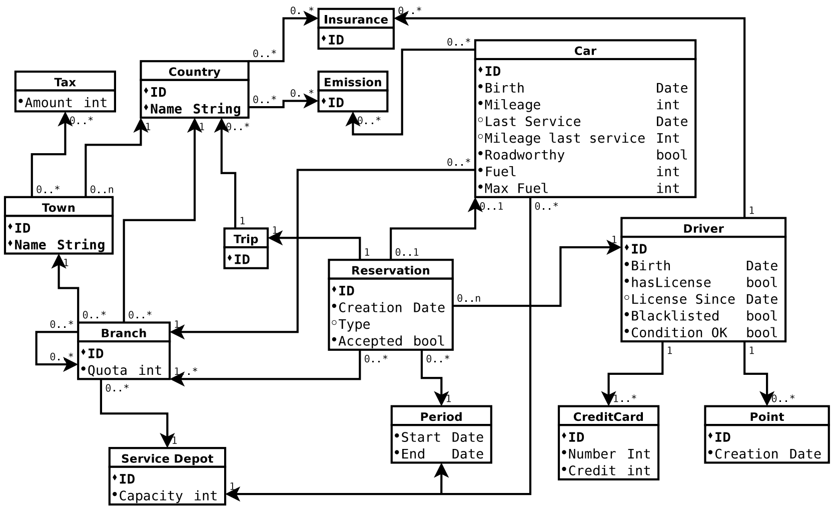

Figure 4.2: Full domain model of the modelled EU-Rent application

interested in modelling the behaviour of a dynamic system, we first consider the static aspect of the system. At every time point, assignments of reservations should satisfy certain invariants. In typical Business Rule solutions, these invariants are left implicit and the programmer should make sure to specify the dynamic behaviour of the system in such a way that the invariants will always hold during the execution of this specification. However, since we use logic as a modelling language, in our approach it is a natural choice to explicitate those invariants and, possibly, use techniques like theorem proving to prove that our dynamic system actually respects those invariants.

A structure of vocabulary SingleState records the actual state of the world (which cars are where, who has requested which reservations, etc.), as well as the current plans of the system (which reservations will be fulfilled by which cars, when should cars be moved between branches, etc.).

The laws of the car scheduling problem that we extracted from the specification can be found in theory ValidState in Figure 4.4. They form a set of invariants of the system. An interesting feature of this theory is that although it specifies conditions that should be satisfied in each individual state of the system, it is still a temporal theory. Indeed, the laws of reservations are temporal in nature, e.g., to state that the same car should not be assigned to two reservations that

$$
\begin{array}{l}
\Sigma_T = \{Car, Driver, Res, Branch, Date\} \\
\Sigma_P = \{ \\
\quad HasLicense(Driver) \\
\quad Accepted(Res) \\
\quad InitLoc(Car, Branch) \\
\quad CarLoc(Car, Date, Branch) \\
\quad Alloc(Res, Car) \\
\quad Overlapping(Res, Res) \\
\quad Move(Car, Branch, Branch, Date) \\
\} \\
\Sigma_F = \{ \\
\quad Today : Date \\
\quad SerialNr(Car) : Int \\
\quad PurchaseDate(Car) : Date \\
\quad Mileage(Car) : Int \\
\quad BirthDate(Driver) : Date \\
\quad StartDate(Res) : Date \\
\quad EndDate(Res) : Date \\
\quad StartLoc(Res) : Branch \\
\quad EndLoc(Res) : Branch \\
\quad Owner(Res) : Driver \\
\}
\end{array}
$$

Figure 4.3: Vocabulary SingleState

overlap in time.

Note that the combination of a structure of vocabulary SingleState and the definitions of predicates *CarLoc* and *Overlapping* can be viewed as a kind of deductive database in which all of the FO axioms in theory ValidState can be evaluated. When an employee manually updates the current state of the system, theory $ValidState$ can be used with the Modelcheck inference (Section 2.4.2). When applied to a structure S representing the current state of the system, this checks if all invariants are still intact.

Dynamic domain knowledge

The modelled EU-Rent system is a dynamic system and as such should contain information on what a valid transition between states is. For such a specification we use a dynamic linear time theory, with *State* used as the type Time (Section 2.3.5) representing how transactions modify the state of the system. A transaction can be any request for a transition in the system, entered by the user, such as “Plan this reservation”, “Move this car”, . . . To express this theory, we will need a new vocabulary SequenceOfStates (Figure 4.5), such that each structure for this vocabulary now corresponds to an entire sequence of states, or equivalently: a transcription of a run of the system.

Using vocabulary SingleState, this can be derived in a principled way by introducing an additional type State and then adding a state argument to all predicates and functions that may change due to transactions. For the EU-Rent application in Figure 4.1, these are the functions and predicates that are drawn in red in Figure 4.1. This new vocabulary allows us to model our system over a series of states.

Dynamic linear time theory ValidRun (Figure 4.6) over vocabulary SequenceOf-States is derived from theory ValidState, since the invariants of ValidState are needed at each state. This is achieved by taking theory ValidState, rephrasing it in the vocabulary SequenceOfStates with a state argument s and quantifying s universally. In the resulting theory, the scheduling predicate *Alloc* has no definitional rules. This means that given an input (a set of reservations, cars and other data) there is not necessarily a fixed output for *Alloc*. The values it can have are merely constrained by the theory. This means that multiple solutions are possible where *Alloc* has a different interpretation.

While ValidRun has no constraints over the transitions between states, it satisfies all requirements to be a LTC theory. It is, however, a trivial LTC theory, since (as long as no constraints on the transitions are added) it does not in fact contain any more information than theory ValidState. In Section 4.1.3, we show how both theories can be used to reason about this system and solve problems and discuss advantages and disadvantages of both approaches.

//Relation Constraints
$\forall r : \exists \leq 1\ c : \mathit{Alloc}(r, c).$
$\forall c\ d : \#\{b_1\ b_2 : \mathit{Move}(c, b_1, b_2, d)\} \leq 1.$
$\forall c\ b\ d : \neg \mathit{Move}(c, b, b, d).$

//Preconditions for accepting a reservation:
$\forall r : \mathit{Accepted}(r) \Rightarrow \exists c[\mathit{Car}] : \mathit{Alloc}(r, c).$
$\forall r\ c : \mathit{Alloc}(r, c) \Rightarrow \mathit{CarLoc}(c, \mathit{StartDate}(r), \mathit{StartLoc}(r)).$
$\forall r : \mathit{Accepted}(r) \Rightarrow \mathit{HasLicense}(\mathit{Owner}(r)).$

//Maximum one reservation at a time per Driver
$\forall r1\ r2 : \mathit{Overlapping}(r1, r2) \wedge r1 \neq r2 \Rightarrow \neg \mathit{Accepted}(r1) \vee \neg \mathit{Accepted}(r2)$
$\quad \vee \mathit{Owner}(r2) \neq \mathit{Owner}(r2).$

//Maximum one reservation at a time per Car
$\forall r1\ r2\ c1 : \mathit{Overlapping}(r1, r2) \wedge r1 \neq r2 \wedge \mathit{Alloc}(r1, c1)$
$\quad \Rightarrow \neg \mathit{Accepted}(r1) \vee \neg \mathit{Accepted}(r2) \vee \neg \mathit{Alloc}(r2, c1).$

//CarLoc is defined in terms of moves and initial location

$$\left\{\begin{array}{ll}
\forall c\ b_1 : \mathit{CarLoc}(c, \mathit{Today}, b_1) & \leftarrow \mathit{InitLoc}(c, b_1). \\
\forall c\ d\ b_1 : \mathit{CarLoc}(c, d+1, b_1) & \leftarrow \mathit{CarLoc}(c, d, b_1) \wedge \\
& \neg(\exists b_2 : \mathit{Move}(c, b_1, b_2, d)) \\
& \wedge \neg \exists r : \mathit{StartLoc}(r) = b_1 \\
& \wedge \mathit{Accepted}(r) \\
& \wedge \mathit{Alloc}(r, c) \wedge \mathit{StartDate}(r) = d) . \\
\forall c\ d\ b_2 : \mathit{CarLoc}(c, d+1, b_2) & \leftarrow \exists b_1 : \mathit{Move}(c, b_1, b_2, d). \\
\forall c\ d\ b_2 : \mathit{CarLoc}(c, d+1, b_2) & \leftarrow \exists r : \mathit{Accepted}(r) \wedge \mathit{Alloc}(r, c) \\
& \wedge \mathit{EndDate}(r) = d \\
& \mathit{EndLoc}(r) = b_2.
\end{array}\right\}$$

//Precondition for the move action
$\forall c\ b_1\ b_2\ d : \mathit{Move}(c, b_1, b_2, d) \Rightarrow \mathit{CarLoc}(c, d, b_1).$

//Definition of Overlapping

$$\left\{\begin{array}{ll}
\forall r_1\ r_2 : \mathit{Overlapping}(r_1, r_2) & \leftarrow (\mathit{StartDate}(r_2) \leq \mathit{EndDate}(r_1) \\
& \wedge \mathit{EndDate}(r_1) \leq \mathit{EndDate}(r_2)) \\
& \vee (\mathit{StartDate}(r_1) \leq \mathit{EndDate}(r_2) \\
& \wedge \mathit{EndDate}(r_2) \leq \mathit{EndDate}(r_1)).
\end{array}\right\}$$

Figure 4.4: The scheduling theory ValidState

$$
\begin{array}{l}
\Sigma_T = \{Car, Driver, Res, Branch, Date, State\} \\
\Sigma_P = \{ \\
\quad HasLicense(Driver) \\
\quad Accepted(Res, State) \\
\quad InitLoc(Car, Branch) \\
\quad CarLoc(Car, Date, Branch, State) \\
\quad Alloc(Res, Car, State) \\
\quad Overlapping(Res, Res) \\
\quad Move(Car, Branch, Branch, Date) \\
\} \\
\Sigma_F = \{ \\
\quad Today : Date \\
\quad Z : State \\
\quad Next(State) : State \\
\quad SerialNr(Car) : Int \\
\quad PurchaseDate(Car) : Date \\
\quad Mileage(Car) : Int \\
\quad BirthDate(Driver) : Date \\
\quad StartDate(Res) : Date \\
\quad EndDate(Res) : Date \\
\quad StartLoc(Res) : Branch \\
\quad EndLoc(Res) : Branch \\
\quad Owner(Res) : Driver \\
\}
\end{array}
$$

Figure 4.5: Vocabulary SequenceOfStates

//Relation Constraints
$\forall r\ s : \exists \leq 1\ c : Alloc(r, c, s).$
$\forall c\ s : \exists \leq 1\ r : Alloc(r, c, s).$
$\forall c\ d : \#\{b_1\ b_2 : Move(c, b_1, b_2, d)\} \leq 1.$
$\forall c\ b\ d : \neg Move(c, b, b, d).$

//Preconditions for accepting a reservation:
$\forall r\ s : Accepted(r, s) \Rightarrow \exists c[Car] : Alloc(r, c, s).$
$\forall r\ c\ s : Alloc(r, c, s) \Rightarrow CarLoc(c, StartDate(r), StartLoc(r), s).$
$\forall r\ s : Accepted(r, s) \Rightarrow HasLicense(Owner(r)).$

//Maximum one reservation at a time per Driver
$\forall r1\ r2\ s : Overlapping(r1, r2) \wedge r1 \neq r2 \Rightarrow \neg Accepted(r1, s) \vee \neg Accepted(r2, s)$
$\quad \vee Owner(r2) \neq Owner(r2).$

//CarLoc is defined in terms of moves and initial location

$$\left\{ \begin{array}{rl} \forall c, b_1, s : CarLoc(c, Today, b_1, s) & \leftarrow InitLoc(c, b_1). \\ \forall c, d, b_1, s : CarLoc(c, d+1, b_1, s) & \leftarrow CarLoc(c, d, b_1, s) \\ & \wedge \neg(\exists b_2 : Move(c, b_1, b_2, d)) \\ & \wedge \neg \exists r : StartLoc(r) = b_1 \\ & \wedge Accepted(r, s) \\ & \wedge Alloc(r, c, s) \wedge StartDate(r) = d. \\ \forall c, d, b_2, s : CarLoc(c, d+1, b_2, s) & \leftarrow \exists b_1 : Move(c, b_1, b_2, d). \\ \forall c, d, b_2, s : CarLoc(c, d+1, b_2, s) & \leftarrow \exists r : Accepted(r, s) \wedge Alloc(r, c, s) \\ & \wedge EndDate(r) = d + 1 \\ & \wedge EndLoc(r) = b_2. \end{array} \right\}$$

//Precondition for the move action
$\forall c\ b_1\ b_2\ d\ s : Move(c, b_1, b_2, d) \Rightarrow CarLoc(c, d, b_1, s).$

//Definition of Overlapping

$$\left\{ \begin{array}{rl} \forall r1\ r2 : Overlapping(r1, r2) & \leftarrow (StartDate(r_2) \leq EndDate(r_1) \\ & \wedge EndDate(r_1) \leq EndDate(r_2)) \\ & \vee (StartDate(r_1) \leq EndDate(r_2) \\ & \wedge EndDate(r_2) \leq EndDate(r_1)). \end{array} \right\}$$

Figure 4.6: The dynamic theory ValidRun

Intermezzo: A rule-driven approach for scheduling reservations

Business Rule systems use another declarative approach to schedule reservations, the first usecase in this project. If-Then rules are used to implement an algorithm that will decide whether cars are available, and if so, will update the current reservation planning. The Business Rule engine runs on a rule base and a working memory that includes the current state and the new request. In our car rental example, the addition of a fact representing a reservation request might trigger rules in such a way that the resulting information allows or prevents the reservation's acceptance.

Using a dynamic vocabulary RentalRequest (Figure 4.7) we study a rule-based approach, using the IDP system on a theory that mimics a Business Rules solution to scheduling reservations. We define a theory ProcessRequest (Figure 4.8) to schedule reservations. Vocabulary RentalRequest contains some extra predicates that give additional input information, such as a partial function $ReqRes(State) : Reservation$, that outputs the request that has to be processed, and some auxilary symbols such as $Candidate(State) : Car$, for readability.

Business Rules often implement a sort of logic rules and their operational semantics corresponds to a sort of bottom up evaluation of rules similar to the computation of intentional predicates in deductive databases. This is of course not always true but it was the case in this use case. Not surprisingly then, we could easily turn these rules into definitional rules of FO(·). For example, a typical Business Rule could be that a reservation can be accepted if a car is available for the required period, and a car is available if it is not allocated to

$$
\begin{array}{l}
\Sigma_T = \{Car, Driver, Res, Branch, Date, State\} \\
\Sigma_P = \{ \\
\quad Available(Car, Reservation, State) \\
\quad Accepted(Res, State) \\
\quad Alloc(Res, Car, State) \\
\quad Overlapping(Res, Res) \\
\} \\
\Sigma_F = \{ \\
\quad Candidate(State) : Car \\
\quad ReqRes(State) : Res \\
\quad StartDate(Res) : Date \\
\quad EndDate(Res) : Date \\
\}
\end{array}
$$

Figure 4.7: Vocabulary RentalRequest

overlapping reservations. If cars are available, another rule would assign an available one for the reservation (the smallest in some order). The definition in Figure 4.8, captures this logic but in a declarative way through definitions.

$$\left\{\begin{array}{ll}
\forall c\ r\ s : \mathit{Available}(c, r, s) & \leftarrow \forall r2 : r \neq r2 \land \mathit{Alloc}(r2, c, s) \\
& \quad \Rightarrow \neg \mathit{Overlapping}(r, r2). \\
\forall c\ s : \mathit{Candidate}(s + 1) = c & \leftarrow \mathit{Available}(c, \mathit{ReqRes}(s), s + 1) \land \\
& \quad (\forall c2 : c2 < c \\
& \quad \Rightarrow \neg \mathit{Available}(c2, \mathit{ReqRes}(s), s + 1)). \\
\forall c\ s : \mathit{Alloc}(\mathit{ReqRes}(s), c, s + 1) & \leftarrow c = \mathit{Candidate}(s + 1). \\
\forall r\ c\ s : \mathit{Alloc}(r, c, s + 1) & \leftarrow \mathit{Alloc}(r, c, s) \land r \neq \mathit{ReqRes}(s). \\
\forall s : \mathit{Accepted}(\mathit{ReqRes}(s), s) & \leftarrow \exists c : \mathit{Alloc}(\mathit{ReqRes}(s), c, s). \\
\forall r1\ r2 : \mathit{Overlapping}(r1, r2) & \leftarrow \mathit{StartDate}(r2) \leq \mathit{EndDate}(r1) \\
& \qquad \lor \mathit{StartDate}(r1) \leq \mathit{EndDate}(r2).
\end{array}\right\}$$

Figure 4.8: Theory ProcessRequest

This rule-based approach has a clear operational semantics. An advantage of our declarative approach over this rule-base approach is that theory ValidRun is a much more natural modelling. While theory ProcessRequest models a more simple scenario, it is already a more complicated theory. When changes are made (for example, the company introduces two classes of cars that can be requested and a reservation can be upgraded if no more cars in the requested class are available), the whole definition has to be modified to make this possible. Adaptability is more straightforward in a theory that only contains the constraints to which any schedule must comply. The second advantage is guaranteed correctness. Provided that the theory ValidState is correct, ValidRun will be correct and it is implicit to the approach that any model of that theory (or equivalently: run of the system) is correct.

4.1.3 Solving Tasks of EU-Rent

Scheduling reservations

The rule-based approach is made with the task of scheduling reservations in mind. Theory ProcessRequest can be used to execute a reservation request transaction, using the progression inference as described in Section 2.4.2.

Note that declarative theories as ValidRun and ValidState can be used for many more tasks than just processing transactions: Given a situation of the system (transcribed by a structure S over vocabulary SingleState) of our system, we can use theory ValidState to see if it this is a valid state of the system, or given a run (transcribed by S' over SequenceOfStates) of our system, we use ValidRun to see if it was a valid run (i.e. if S' is a model of theory ValidRun). Theory ValidRun can also be used for simulation, for example, we can add transactions (for example: I move this car to the other branch and then accept these five reservations) and check if there exists a valid run of the system, that starts in the current structure, where those transactions are processed.

We can also use these theories to run a system that can process reservations or other transactions. Given a structure S representing the current state, we can manually modify the structure, adding this reservation to the domain of reservations and performing a modelexpansion inference to find a new structure S' representing a valid state of the system that contains the new reservation. This method seems to perform the task well, especially since there are no extra constraints on the transition between states. However, it is difficult to generalise and it leaves the knowledge about the state transition completely implicit. Another method is using the progression inference on theory ValidRun and the modified structure, which also calculates a new structure satisfying the theory. This method does allow to work with constraints on transitions between states.

An operational difference with the rule-based approach is that the result when using ValidRun or ValidState is not deterministic: at each state, the system can freely choose a new schedule. This may pose practical problems for the company in the sense that planning of manpower and resources becomes very difficult (e.g. will it be necessary to move a car from one branch to another?). A solution for this problem would be to compute solutions minimising an objective function that penalises modifications to the previous schedule. This contrasts sharply with the rule driven approach, where a deterministic description of a next state is specified. In practice, the main concern of a business manager is to make profit. Therefore, such a manager will not care which specific next state is selected by a system as long as the decisions made by the system are profitable. This is in fact a search problem, where we want to maximise a certain profit function, namely, the income we obtain from renting out cars minus the sum of all expenses. If it is possible to fulfill 8 new reservations by cancelling 1, it is possible that a company might want to do that.

In fact the ultimate goal is, given a current state of the system and some new reservations, to calculate a new optimal state of the system: a new optimal plan incorporating new reservations, accepting as many as possible, with making as few changes as possible. This new state should definitely be a valid state, hence it has to satisfy theory ValidState.

In the rule-based approach, manual heuristics are used as an attempt to maximise the profit. For example, a rule might be "preferably assign a free car to a new reservation, only if that is impossible, try shifting the schedule". A disadvantage of the rule based approach is that the used optimality criterion is left implicit and not all cases can be covered, since the optimalisation is among others limited by the creativity of the programmer.

In this section, we will describe more fundamental approaches. One using ValidState and a *Weighted Model Revision inference.*

Definition 4.1.1. *ReviseWeighted(T,S,f)* is a derived inference with as input a structure S and theory T over the same vocabulary and a partial function f from the set of predicates and functions to pairs of integers. Output is a structure S' that is a model of T, that is only different from S for elements of the vocabulary that are in the domain of f, or $UNSAT$ if there exists no such structure. The resulting structure S' has a minimal value for *cost*, where:

$$cost = \sum_{\sigma \in Dom_f} (f(\sigma))_1 * \#\{\overline{d}|\overline{d} \in \sigma^{S'} \wedge \overline{d} \notin \sigma^{S}\}$$

$$+ \sum_{\sigma \in Dom_f} (f(\sigma))_2 * \#\{\overline{d}|\overline{d} \in \sigma^{S} \wedge \overline{d} \notin \sigma^{S'}\}$$

Intuitively, the function f maps predicates and functions to the cost to make this predicate/function more true and the cost to it make more false. The inference calculates a model for T that only differs from S on some symbols, where making changes to the interpretations of symbols can be encouraged or penalized.

Example 4.1.2. In the situation of the Car Rental company, we might give these arguments to our model revision:

$$CurrentState,\ ValidState,$$
$$\{Accepted \mapsto (-100, 300), CarLoc \mapsto (50, 50), Alloc \mapsto (20, 20)\}$$

This input expresses for example that only Accepted, CarLoc and Alloc can be changed. We also see that it is very good (profit 100) to accept more reservations, while it is costly (cost 300) to cancel a previously confirmed reservation. Of course, these parameters can be fine-tuned depending on real life information. The weighted model revision then calculates a new model: a revision of the old model that minimises the cost (sum over all changed atoms of the weight assigned to that particular change).

A more general approach using theory ValidRun (and thus allowing constraints on state transition) uses *Weighted Progression.*

Definition 4.1.3. *Weighted Progression* is a derived inference with as input a single-state structure S_0 over Σ^{ss}_{LTC}, a theory T over a linear-time vocabulary Σ_{LTC} and a numerical term t. Output is a single-state partial structure S_1 such that $bistate(\Sigma_{LTC}, S_0, S_1) \models T$ such that for every other S'_1 satisfying $bistate(\Sigma_{LTC}, S_0, S'_1) \models T$ it is the case that $t^{S_1} \leq t^{S'_1}$.

A user can add a definition of cost to the ValidRun theory and then use weighted progression to calculate a possible next state that minimises the cost.

Example 4.1.4. To model the same scenario as in example 4.1.2, we add a term $Cost : Int$ to SequenceOfStates and following definition to ValidRun:

$$
\begin{aligned}
&\{\\
&\quad Cost = -100 * \#\{x | \neg Accepted(x, Z) \land Accepted(x, Next(Z))\}\\
&\qquad\qquad + 300 * \#\{x | Accepted(x, Z) \land \neg Accepted(x, Next(Z))\}\\
&\qquad\qquad + 50 * \#\{c\ d\ b | \neg CarLoc(c, d, b, Z) \land CarLoc(c, d, b, Next(Z))\}\\
&\qquad\qquad + 50 * \#\{c\ d\ b | CarLoc(c, d, b, Z) \land \neg CarLoc(c, d, b, Next(Z))\}\\
&\qquad\qquad + 20 * \#\{r\ c | \neg Alloc(r, c, Z) \land Alloc(r, c, Next(Z))\}\\
&\qquad\qquad + 20 * \#\{r\ c | Alloc(r, c, Z) \land \neg Alloc(r, c, Next(Z))\}\\
&\}
\end{aligned}
$$

We use $ProgressWeighted(ValidRun, S_0, Cost)$ with a structure S_0 representing the current state, to calculate a good next state for the system.

To conclude, we studied 4 approaches to schedule reservations. The reservation is added to the structure and one of four inferences can be used:

$Modelexpand(ValidState, S_0)$ Calculates a model that contains the added reservation and satisfies all constraints.	$ReviseWeighted(ValidState, S_0, f)$ Calculates a model that contains the added reservation and satisfies all constraints where the cost associated with f is minimized.
$Progress(ValidRun, S_0)$ Calculates a next step such that the combination of both steps satisfies the bistate theory.	$ProgressWeighted(ValidRun, S_0, t)$ Calculates a next step with minimal cost t such that the combination of both steps satisfies the bistate theory.

It is clear that the bottom-right approach is the most general and extensible, since it allows to state constraints on state transition and to do optimisation. The top-left approach is the most simple. We continue with using the $ProgressWeighted$ approach in the rest of this text.

4.1.4 Adding New objects

Our second use case is an extension of the first one: we want to extend theories with the possibility of adding new objects. In many software systems, adding a new object is a trivial task: new entries in a database are constantly being created. The car rental problem domain also allows for cars to be purchased, and hence it must be possible to add a new car to the set of cars. In logic speak: it must be possible to add a new object to the Car domain.

The obvious way to do this would simply be to extend the structure (or database) with an extra car. However, this means that the knowledge about the introduction of a new car is introduced is not present in the logical theory representing the problem domain. This also means that information about this transaction is not reusable for other forms of reasoning, e.g. making simulations of the system, verification, etc. To remedy this, we propose a new logical operator representing the knowledge that a new domain element is created.

Before we explain the exact syntax and semantics of this operator in Section 4.2, we take a look at how the addition of a novel logic operator could solve the creation of a car in our car rental problem domain:

$$\left\{\begin{array}{l} \forall s\ sn:\ \textbf{new}\ c:\ SerialNr(c) = sn \\ \qquad \land PurchaseDate(c) = s \land Mileage(c) = 0 \leftarrow BuyCar(sn, s). \end{array}\right\}$$

This definition states that if a car with serial number sn is bought in a certain state s, a new car has to be added to the system with that specific serial number, purchased at that point in time, with zero mileage, and some other initial attributes. Note that we added a predicate $BuyCar(Int, State)$ to the vocabulary SequenceOfStates, representing the serial numbers of any cars bought. In order to support this kind of expressions, we introduce a new logic, FO($\cdot^+$), which basically extends FO($\cdot$) by allowing more complex formulas in the head of definitions.

4.2 FO($\cdot^+$): a logic accomodating a "new"-operator

4.2.1 Syntax

An FO($\cdot^+$) specification consists of a set of sentences φ in FO and a set of extended definitions: definitions with a FO sentence in the body, but an extended notion of a head of a definition.

Definition 4.2.1. A formula φ_h is a *head formula* if

- φ_h is an atom
- $\varphi_h = \varphi_1 \land \varphi_2$ with φ_1 and φ_2 head formulas.
- $\varphi_h = \textbf{new}\ \psi$ with ψ a head formula.

Definition 4.2.2. An *extended definition* Δ over $\mathcal{L}_{FO}$ is a set of rules of the form

$$\forall \overline{x}:\ \varphi_h \leftarrow \varphi_b$$

where $\varphi_b \in \mathcal{L}_{FO}$, φ_h is a *head formula* and all the free variables of φ_b and φ_h are among the variables in $\overline{x}$.

4.2.2 Intuitions

Like in informal definitions, any rule produces a set of atoms described by the head formula. Even though conjunctive head formulas resemble simple

FO formulas the intended meaning of an extended rule is **not** “if a body is true, the corresponding head formula is true”. It means that if the body of a certain rule is true, the head formula will determine a set of atoms to be true (and nothing more). The reason for this is that inductive definitions describe a process that starts from an empty interpretation for defined symbols and keeps executing rules (or deriving that certain heads can never be derived anymore) until fixpoint. In the extended formalism, one rule can make several atoms true at once (and create new elements). Conjunctive head-rules represent multiple effects of the same action: multiple things becoming true together. Hence, the set of atoms made true is simply the union of the atoms made true by subformulas. The **new** operator causes the creation of a new domain element different from all other elements of this same type. These are the intuitions how such rules should be read. We now describe a formal semantics for this logic that preserves the intuitions described in this section.

4.2.3 Semantics

We define a transformational semantics for FO($\cdot^+$): a transformation from a specification in FO($\cdot^+$) to a specification in FO($\cdot$) with an explanation of how this transformation preserves the intuitions from the previous section.

Given a vocabulary Σ used for writing a theory T in FO($\cdot^+$), the FO($\cdot$) translation of T will be written in a slightly modified version of T.

Definition 4.2.3. Given a vocabulary Σ and a theory T that we want to translate, we define a vocabulary Σ_T as:

- Σ_T contains all elements of Σ
- For each defined type S in T, we add a new predicate $InType_S$, with arity one and type S.
- For each **new** rule of the form

  $$\forall \overline{x}[\overline{t}] : \mathbf{new}\ y[s] : \varphi_h(\overline{x}, y) \leftarrow \varphi_b$$

  we introduce a new partial function $F : \overline{t} \rightarrow s$ (for every “new-rule”, this is a new function).
- For every **new** rule as above, we add a predicate $Dom_F(\overline{t})$ representing the domain of each F.

The translation theory $T' = Trans(T)$ in FO($\cdot$) contains all formulas of T. For the definitions we define a recursive translation that translates a rule into one or more rules with a strictly smaller head formula. We recursively apply this transformation until all rules have simple heads (atoms).

- The simplest rules to transform are conjunctive rules. Since they represent multiple effects of the same action, they can simply be translated to multiple rules. A rule
$$(\varphi_1 \land \varphi_2) \leftarrow \psi$$
is translated as
$$\varphi_1 \leftarrow \psi$$
$$\varphi_2 \leftarrow \psi$$

- A **new** rule (as above) is translated into three rules:
$$\forall \bar{x} : \text{Dom}_F(\bar{x}) \leftarrow \varphi_b.$$
$$\forall \bar{x}, y : \text{InType}_s(y) \leftarrow \text{Dom}_F(\bar{x}) \land F(\bar{x}) = y.$$
$$\forall \bar{x}, y : \varphi_h(\bar{x}, y) \leftarrow \text{Dom}_F(\bar{x}) \land F(\bar{x}) = y.$$
The intuitions behind these rules are the following: by doing this for every rule containing creator symbols, we get that InType_s is defined as the union of all elements that occur as image of some creation function. I.e. *InType* will be true exactly for all those elements such that a new-rule gets triggered. Furthermore, for those elements, φ_h should hold.

Since newly created elements are completely unique, we should add unique name axioms (UNA) for the created elements:

- Every creatorfunction is injective: $\forall \bar{x}, \bar{y} : F(\bar{x}) = F(\bar{y}) \Rightarrow \bar{x} = \bar{y}$.
- Elements created by different creatorfunctions are unique: $\forall \bar{x} : F_1(\bar{x}) \neq F_2(\bar{x})$

The created elements are also a closed domain: the number of new elements are exactly those elements that are "created" by a certain constructing rule. We also want to establish the link between the domain predicate and the corresponding partial function: creatorfunctions are defined on (and only on) their domain. Hence we add the following constraints to our theory:

- $\forall x[S] : \text{InType}_S(x)$,

- $\forall \bar{x} : \mathrm{Dom}_F(\bar{x}) \Leftrightarrow \exists y[s] : F(\bar{x}) = y.$

No modifications on structures are made.

Example 4.2.4. We illustrate this translation on the above example. Given vocabulary SingleState and the rule creating the car in FO($\cdot^+$):

$$\left\{ \begin{array}{l} \forall s\ sn : \ \textbf{new}\ c : \ SerialNr(c) = sn \\ \qquad \wedge PurchaseDate(c) = s \wedge Mileage(c) = 0 \leftarrow BuyCar(sn, s). \end{array} \right\}$$

We look at what we add to vocabulary $SequenceOfStates$ and we look at translation theory T' that models the same behaviour. The vocabulary contains 3 new symbols:

1. $InType_{Car}$: a predicate representing all created cars
2. F: a partial function that maps serial numbers and purchase dates to new cars.
3. Dom_F: a predicate representing the domain of F.

A theory containing the rule creating the new car will be modified as:

$$\left\{ \begin{array}{ll} \forall s\ sn : \ Dom_F(s, sn) & \leftarrow BuyCar(sn, s). \\ \forall s\ sn : \ c : \ InType_{Car}(c) & \leftarrow Dom_F(s, sn) \wedge F(sn, s) = c. \\ \forall s\ sn : \ c : \ SerialNr(c) = sn & \\ \qquad \wedge PurchaseDate(c) = s & \\ \qquad \wedge Mileage(c) = 0 & \leftarrow Dom_F(s, sn) \wedge F(sn, s) = c. \end{array} \right\}$$
$$\forall s_1\ sn_1\ s_2\ sn_2 : F(s_1, sn_1) = F(s_2, sn_2) \Rightarrow s_1 = s_2 \wedge sn_1 = sn_2.$$
$$\forall c[Car] : \mathrm{InType}_{Car}(c).$$
$$\forall s\ sn : \mathrm{Dom}_F(s, sn) \Leftrightarrow \exists c[Car] : F(s, sn) = c.$$

4.3 Reasoning with FO($\cdot^+$) in a finite domain reasoning system

The above transformation cannot be applied in a context when reasoning with a finite domain. It assumes that the interpretations of the types can be unknown when finding models for a theory. This assumption is not valid for the IDP system, hence another transformation had to be implemented. Below we show

how a modelexpansion inference can be done for a FO($\cdot^+$) theory in a reasoning system with a finite domain solver.

The implementation of this semantics uses an overapproximation. It first replaces all defined types with a overapproximation (a set of potential objects, if you will). We remove the constraint that all elements of a type S are in the predicate $InType_S$ and use this predicate as the subset of elements of S that are really created. Since the types are overapproximated, we have to be careful: all quantification over defined types should become conditional. This means we replace any universal quantification $\forall \bar{x}[S] : \varphi(\bar{x})$ by $\forall \bar{x}[S] : InType_S(\bar{x}) \Rightarrow \varphi(\bar{x})$ and any existential quantification $\exists \bar{x}[S] : \varphi(\bar{x})$ by $\exists \bar{x}[S] : InType_S(\bar{x}) \land \varphi(\bar{x})$. After the solving step, we redefine the interpretation of type by the subset in the interpretation of $InType_S$. The projection of this structure to the original vocabulary Σ, we get a structure that satisfies all constraints and where the new elements have been created.

4.3.1 Using FO($\cdot^+$) in the car rental application

Processing a transaction where a new car is bought was the motivation behind developing FO($\cdot^+$). We give an overview how given a structure S over Vocabulary $SingleState$ state representing the current state of the system and a theory T containing theory SequenceOfStates and containing the rule creating a new car can be used to create a structure S' satisfying T, having a new car. Vocabulary $SequenceOfStates$ is a linear-time vocabulary (Definition 2.3.3), with $State$ as the type $Time$, and Vocabulary $SingleState$ is the derived single state vocabulary of $SequenceOfStates$ (Definition 2.3.6). This means that we can use the progression inference $Progress(T, S)$ for a FO($\cdot$) theory T over $SequenceOfStates$ and a structure S over $SingleState$. We define a generalisation of the progression inference defined in Section 2.4.2 that can be used on FO($\cdot^+$) theories.

Definition 4.3.1. Progress(T, S_0) is a logical inference with input:

- a FO($\cdot^+$) theory T over a linear-time vocabulary Σ_{LTC}
- a single-state structure S_0 over Σ_{LTC}^{ss}

And output: The projection to vocabulary Σ_{LTC}^{ss} of a single-state partial structure S_1 over FO($\cdot^+$) such that $bistate(\Sigma_{LTC}, S_0, S_1) \models Trans(T)$.

4.4 Context and Future Work

The work presented in this chapter was a preliminary study of the problem setting in modelling Business Rule applications in our KBS, and the motivation for further research. It led to the FO(C) logic from [Bogaerts et al., 2014c] and [Bogaerts et al., 2014b]. To give an overview of the context of the work in this chapter and the future work it led to we will discuss this FO(C) logic.

FO(C) is a combination of C-Log and first-order logic, and is a member of the FO(·) family of languages. The goal of C-Log is to capture complex causal relations in a knowledge representation language. It is a generalisation of *CP-logic* [Vennekens et al., 2009], that allows to express

- Non-deterministic choices between a dynamic set of alternatives.
- The causation of the creation of new objects.
- The nesting of causal laws: a entire causal law can be caused in itself.

The creation of new objects was already studied in this chapter. Similar constructs were developed in various extensions of Datalog [Abiteboul and Vianu, 1991, Van den Bussche and Paredaens, 1995]. In the LogicBlox system they have a notion of implicit existentially quantified head variables, that correspond with the **new**-expression discussed above.

4.5 Conclusion

We evaluated the Knowledge Base Paradigm and compared it to Business Rules, using a real life application from the Business Rules domain: the EU-Rent Car Rental company. We looked at two specific use cases in this application: the scheduling of reservations and the task of adding new cars to the system.

The main advantage of the KB approach over rule-based systems is that a specification in a KBS is an explicit modelling of the domain knowledge. This allows for easy modification when the domain knowledge changes and reuse of knowledge and specifications in various inferences. In this chapter we have used the same specification of domain knowledge for verification, simulation and execution.

To study the difference between a rule-based approach and our KBS in more detail, we modelled a rule-based specification using the definitional rules of

FO(·) and used it to schedule reservations. We noticed another practical advantage of the KB approach. The determinism that is implicit in such a rule-based approach does not always allow the system to optimise profit; it encodes a heuristic to deterministically pick a quite good solution, and excludes other, possibly better solutions.

We discussed a new derived inference that allows us to revise models where changes to certain interpretations can be encouraged or penalised. This new inference made that we could revise a schedule, that introduced no unneeded changes.

The current version of the IDP system and FO(·) language were not fit for making small database changes, such as adding a new car to the domain. These lacks were no fundamental faults of the KB paradigm, but rather a lack of expressivity of the current version of the language we used. Therefore, we described a new logic, using an extension of the definitions in FO(·) which filled the gap that was left in our modelling of a car rental system.

5 A knowledge representation language for reasoning in a distributed environment: dAEL

Access control is concerned with methods to determine which principal (i.e. user or program) has the right to access a resource, e.g., the right to read or modify a file. Multiple logics have been proposed for distributed access control [Abadi, 2003, Gurevich and Neeman, 2008, Abadi, 2008, Garg and Pfenning, 2012, Genovese, 2012]. Most of these logics use a modality $k\,\textit{says}\,\varphi$ indexed by an agent k. Logics of *says*-based access control are designed for systems in which different agents can issue statements that become part of the access control policy. $k\,\textit{says}\,\varphi$ is usually rendered as "k supports φ", which can be interpreted to mean that k has issued statements that – together with some additional information present in the system – imply φ. Different access control logics vary in their account of which additional information may be assumed in deriving the statements that k supports.

In Section 5.4, we argue that it is reasonable to assume that the statements issued by an agent are a complete characterization of what the agent supports. This is similar to the motivation behind AEL to consider an agent's theory to be a complete characterization of what the agent knows [Moore, 1985, Levesque, 1990, Niemelä, 1991, Denecker et al., 2011]. This motivates an application of

AEL to access control. However, AEL is designed to model the knowledge of one single (introspective) rational agent. An extension to AEL with multiple agents has been defined by Vlaeminck et al. [2012], but this extension requires a global stratification on the agents, which is undesirable for a distributed system. Therefore, we extend AEL to a distributed setting, where each agent has his own policy, possibly referring to the policy of others. We argue in Section 5.4 that the proposed extension provides a good formal model of the *says*-modality.

As the term "autoepistemic logic" suggests, AEL was designed to model (a single agent's) knowledge, including knowledge derived from reasoning about knowledge. However, the formalism of AEL can be applied to model other modalities too. Note that when making a claim about an agent's knowledge, we make a claim about that agent's internal state of mind. However, the formalism of AEL does not presuppose that its K modality represents an internal state of mind of an agent. In our approach, we use theories of agents to represent their publicly announced access control policy. In this case, we can interpret the K modality to refer to the public commitments of an agent, i.e., interpret $K\phi$ to mean that the agent in question has publicly made statements that imply ϕ. In line with the terminology in the AEL literature, we use the terms "knowledge" and "belief" (and the verbs "knows" and "believes") interchangeably to refer to the K modality of AEL, without thereby implying that it represents an internal state of mind. Similarly, we use terms like "positive introspection into other agents' knowledge" to describe a certain formal behaviour of the modality, without implying that an agent knows the internal state of mind of another agent.

This chapter proposes an extension of autoepistemic logic AEL [Moore, 1985] called Distributed Autoepistemic Logic (dAEL) with multiple agents with full introspection into each other's knowledge. In dAEL, we assume agents to have full (positive and negative) introspection into other agents' knowledge. This is of course an unreasonable assumption when the K modality represents an internal state of mind like actual knowledge. It is, however, reasonable when the $K\phi$ is interpreted to mean that an agent has issued statements that imply ϕ.

In Section 5.1, we first define the syntax of dAEL, then define a 2-valued and a 3-valued semantic operator for dAEL, and then show how approximation fixpoint theory can be applied to these operators to define five semantics for dAEL corresponding to five well-known semantics for AEL. In Section 5.2, we define a mapping from dAEL to AEL and show that for a subset of the logic defined by a strong consistency constraint, the mapping preserves the five semantics. In Section 5.4, we argue that dAEL with the well-founded semantics can be fruitfully applied to access control. We conclude this chapter in Section 5.5.

The work presented in this chapter was published as a conference paper at the International Joint Conference on Artificial Intelligence in [Van Hertum et al., 2016a].

5.1 Syntax and Semantics of distributed Autoepistemic Logic

We describe the syntax of dAEL as an extension of FO. Throughout this entire chapter, we assume a fixed set of agents $\mathcal{A}$, vocabulary Σ and a fixed domain D. Furthermore, we assume that for each $d \in D$, d is a constant symbol in dAEL whose interpretation in all structures is d. The class of Σ-structures with domain D (and each $d \in D$ interpreted by itself) represents all potential objective states of affairs.

An epistemic state of an agent is a set of structures of this class of structures. By imposing the use of this class of structures in the representation of the epistemic state of agents, two important constraints on the knowledge of agents are induced:

- Every agent knows the domain of discourse to be D.
- Every agent knows the identity of each constant $d \in D$.

Thus, we do not model situations where one agent has incomplete knowledge of the domain or where different agents have different opinions about the domain.

Next to constants that represent domain elements, we also allow constants (as 0-ary function symbols). Note that there are no constraints on the interpretation of these symbols: agents can have no information about them, or differ in opinion on their interpretation.

5.1.1 Syntax and Basic Semantic Notions

Definition 5.1.1. We define the language $\mathcal{L}_d$ of distributed autoepistemic logic using the standard recursive rules of first-order logic, augmented with:

$$K_A(\psi) \text{ is a formula if } \psi \text{ is a formula and } A \in \mathcal{A}$$

$$\exists A\psi \text{ is a formula if } \psi \text{ is a formula and } A \in \mathcal{A}$$

dAEL generalises autoepistemic logic in the sense that we have multiple agents, each having their own theory describing their belief or knowledge about the world.

Definition 5.1.2. A *distributed theory* is an indexed family $\mathcal{T} = (T_A)_{A \in \mathcal{A}}$ where each T_A is a set of sentences in dAEL. We will denote T_A also as $(\mathcal{T})_A$.

Following example shows how a specification in dAEL looks.

Example 5.1.3. In a university, there are 2 agents: Professor Alice and her postdoctoral researcher Bob. Together they manage 3 students: Charlie, Dylan and Ellen. The professor has two databases Db and $Db2$, for which access needs to be controlled. This is the policy of Alice:

- Charlie gets access to Db, without any restrictions.
- Ellen's access of Db is decided by Bob, she gets access if and only if Bob says so.
- For $Db2$, any student gets access, unless Bob explicitly says otherwise.
- Dylan gets access to any database, unless I have a reason to decide otherwise.

Postdoc Bob has following policy:

- If anyone is explicitly distrusted (does not get access) by the professor, I will follow her and will also not allow access.
- Ellen can have access to Db.
- Ellen cannot have access to $Db2$.

A formalisation of this can be made, using a vocabulary

$$\begin{aligned}
&\Sigma = \{\\
&\quad \Sigma_T = \{student, resource\}\\
&\quad \Sigma_P = \{Access(student, resource)\}\\
&\quad \Sigma_F = \{Charlie, Dylan, Ellen, Db, Db2\}\\
&\}
\end{aligned}$$

We will use A, B, C, D, E to respectively denote *Alice, Bob, Charlie, Dylan* and *Ellen.* In dAEL, the policy is modeled as a distributed theory $\mathcal{T} = (T_A, T_B)$, where

$$
\begin{aligned}
&T_A = \{\\
&\quad Access(C, Db).\\
&\quad Access(E, Db) \Leftrightarrow K_B Access(E, Db).\\
&\quad \forall s[Student] : \neg Access(s, Db2) \Leftarrow K_B \neg Access(s, Db2).\\
&\quad \forall x : \neg K_A \neg Access(D, x) \Rightarrow Access(D, x).\\
&\}\\
&T_B = \{\\
&\quad \forall x,\ y : K_A \neg Access(x, y) \Rightarrow \neg Access(x, y).\\
&\quad Access(E, Db).\\
&\quad \neg Access(E, Db2).\\
&\}
\end{aligned}
$$

In Section 2.5, we introduced possible world structures: sets of structures that are consistent with an agents knowledge. To generalise the concept of possible world structure, we introduce the notion of a Distributed Possible World Structure (DPWS) which represents the knowledge of multiple agents.

Definition 5.1.4. A DPWS $\mathcal{Q}$ is an indexed family $\mathcal{Q} = (Q_A)_{A \in \mathcal{A}}$, where Q_A is a possible world structure for each $A \in \mathcal{A}$. We will denote Q_A also as $(\mathcal{Q})_A$.

The knowledge order of Definition 2.5 can be extended pointwise to DPWS's. One DPWS contains more knowledge than another if each agent has more knowledge: given two DPWS's $\mathcal{Q}_1$ and $\mathcal{Q}_2$, we define $\mathcal{Q}_1 \leq_K \mathcal{Q}_2$ if $(\mathcal{Q}_1)_A \leq_K (\mathcal{Q}_2)_A$ for each $A \in \mathcal{A}$.

The value of a sentence is obtained like in AEL by evaluating each modal operator with respect to the right agent.

Definition 5.1.5. The *value* of a sentence φ in $\mathcal{Q}, I$ (denoted $\varphi^{\mathcal{Q},I}$) is defined inductively by the standard recursive rules for first-order logic, augmented with:

$$(K_A\varphi)^{\mathcal{Q},I} = \mathbf{t} \qquad\qquad \text{if } \varphi^{\mathcal{Q},J} = \mathbf{t} \text{ for each } J \in \mathcal{Q}_A$$

Since the set $\mathcal{A}$ of agents is fixed, we will interpret $\exists A \psi$ as syntactic sugar for the finite disjunction $\bigvee_{A' \in A} \psi[A/A']$.

In order to generalise this valuation to a partial setting, we define a generalisation of belief pairs: a belief pair (i.e., an upper and lowerbound for the possible world structure) for each agent.

Definition 5.1.6. A Distributed Belief Pair (DBP) is an indexed family $\mathcal{B} = (B_A)_{A \in \mathcal{A}}$, where for each $A \in \mathcal{A}$, B_A is a pair (P_A, S_A) of possible world structures. We will denote B_A also as $(\mathcal{B})_A$.

Recall that we call a DBP consistent if $(\mathcal{B})_A$, that is the belief pair associated with agent A, is consistent (Section 2.6.3) for all A . The precision order on DBP's is a pointwise extension of the precision order on belief pairs: $\mathcal{B} \leq_p \mathcal{B}'$ if $(\mathcal{B})_A \leq_p (\mathcal{B}')_A$ for each $A \in \mathcal{A}$. By abuse of notation, we sometimes identify $\mathcal{B}$ with a pair of distributed possible world structures $(\mathcal{B}^c, \mathcal{B}^l)$ where $\mathcal{B}^c$ stands for the tuple $(B^c_A)_{A\in\mathcal{A}}$ of *conservative* bounds (a DPWS), and $\mathcal{B}^l$ for the tuple $(B^l_A)_{A\in\mathcal{A}}$ of *liberal* bounds (a DPWS). Intuitively, for each agent A, B^c_A is the underestimation and B^l_A is the overestimation of the knowledge of A, so B^c_A contains all knowledge the agent certainly has (or, what he knows if we look at his belief pair from a conservative viewpoint) and B^l_A all knowledge the agent possibly has (or, what he knows if we look at his belief pair from a liberal viewpoint).

A DBP $\mathcal{B}$ is more precise than $\mathcal{B}'$ if for each agent A: $\mathcal{B}^c_A \geq_K \mathcal{B}'^c_A$ and $\mathcal{B}^l_A \leq_K \mathcal{B}'^l_A$. The following proposition follows easily from the equivalent result in AEL.

Proposition 5.1.7. *The set of all DPWS's forms a complete lattice when equipped with the order $\leq_K$. The set of all DBP's forms a lattice when equipped with the order $\leq_p$. The latter is the bilattice of the former.*

As before, we assume that all DBP's are consistent.

The notion of three-valued valuations is extended to the distributed setting by evaluating each modal operator with respect to the correct agent.

Definition 5.1.8. The *value* of φ with respect to DBP $\mathcal{B}$ and interpretation I (notation $\varphi^{\mathcal{B},I}$) is defined inductively by replacing the fifth rule in the recursive definition of the three-valued valuation of an AEL formula (Definition 2.6.2) by:

$$(K_A\varphi)^{\mathcal{B},I} = \begin{cases} \mathbf{t} & \text{if } \varphi^{\mathcal{B},I'} = \mathbf{t} \text{ for all } I' \in (\mathcal{B}^c)_A \\ \mathbf{f} & \text{if } \varphi^{\mathcal{B},I'} = \mathbf{f} \text{ for some } I' \in (\mathcal{B}^l)_A \\ \mathbf{u} & \text{otherwise} \end{cases}$$

When the DBP is exact, the valuation corresponds with the two-valued valuation defined in Definition 5.1.5.

Proposition 5.1.9. *Given a DPWS $\mathcal{Q}$, a structure I, and a dAEL-formula φ, we have that*

$$\varphi^{\mathcal{Q},I} = \varphi^{(\mathcal{Q},\mathcal{Q}),I}$$

Proof. We prove this by induction on the structure of φ. The only non-trivial case is the one where $\varphi = K_A\psi$ for an agent $A \in \mathcal{A}$. In this case:

$$(K_A\psi)^{(\mathcal{Q},\mathcal{Q}),I} = \begin{cases} \mathbf{t} & \text{if } \psi^{(\mathcal{Q},\mathcal{Q}),I'} = \mathbf{t} \text{ for all } I' \in ((\mathcal{Q},\mathcal{Q})^c)_A \\ \mathbf{f} & \text{if } \psi^{(\mathcal{Q},\mathcal{Q}),I'} = \mathbf{f} \text{ for some } I' \in ((\mathcal{Q},\mathcal{Q})^l)_A \\ \mathbf{u} & \text{otherwise} \end{cases}$$

$$= \begin{cases} \mathbf{t} & \text{if } \psi^{\mathcal{Q},I'} = \mathbf{t} \text{ for all } I' \in (\mathcal{Q})_A \\ \mathbf{f} & \text{if } \psi^{\mathcal{Q},I'} = \mathbf{f} \text{ for some } I' \in (\mathcal{Q})_A \\ \mathbf{u} & \text{otherwise} \end{cases}$$

$$= \begin{cases} \mathbf{t} & \text{if } \psi^{\mathcal{Q},I'} = \mathbf{t} \text{ for all } I' \in (\mathcal{Q})_A \\ \mathbf{f} & \text{if } \psi^{\mathcal{Q},I'} = \mathbf{f} \text{ for some } I' \in (\mathcal{Q})_A \end{cases}$$

$$= (K_A\psi)^{\mathcal{Q},I}$$

where the first steps follows from the induction hypothesis, and the second by the observation that the valuation $(\cdot)^{\mathcal{Q},I}$ is two-valued (as in: the valuation $\varphi^{\mathcal{Q},I}$ for any dAEL formula φ). □

The valuation essentially provides us with the means to apply AFT to lift the class of semantics of AEL to dAEL.

5.1.2 Semantics for dAEL

The two- and three-valued valuations form the building blocks to extend the semantic operator and its approximator from AEL to dAEL. This will allow us to apply AFT and lift all the various semantics defined for AEL to dAEL. In order to generalise the various semantics for AEL to the distributed setting, we first define a semantic operator $\mathcal{D}_\mathcal{T}$.

Definition 5.1.10. The knowledge revision operator for a distributed theory $\mathcal{T} = (T_A)_{A\in\mathcal{A}}$ is a mapping on the set of distributed possible world structures, defined by

$$\mathcal{D}_\mathcal{T}(\mathcal{Q}) = (\{I \mid (T_A)^{\mathcal{Q},I} = \mathbf{t}\})_{A\in\mathcal{A}}$$

This revision operator revises the knowledge of all agents simultaneously, given their current states. Fixpoints of this operator represent states of knowledge of the agents that cannot be revised any further. Or, in other words, distributed possible world structures that are consistent with the theories of all agents. AFT studies fixpoints of semantic operator $\mathcal{D}_\mathcal{T}$ using fixpoints of an approximator $\mathcal{D}^*_\mathcal{T}$.

Definition 5.1.11. The approximator for a distributed theory $\mathcal{T} = (T_A)_{A\in\mathcal{A}}$ on a DBP $\mathcal{B}$ is defined by $\mathcal{D}^*_{\mathcal{T}}(\mathcal{B}) = (\mathcal{D}^c_{\mathcal{T}}(\mathcal{B}), \mathcal{D}^l_{\mathcal{T}}(\mathcal{B}))$, where

$$\mathcal{D}^c_{\mathcal{T}}(\mathcal{B}) = (\{I \mid (T_A)^{\mathcal{B},I} \neq \mathbf{f}\})_{A\in\mathcal{A}}$$

$$\mathcal{D}^l_{\mathcal{T}}(\mathcal{B}) = (\{I \mid (T_A)^{\mathcal{B},I} = \mathbf{t}\})_{A\in\mathcal{A}}$$

Proposition 5.1.12. *The approximator is $\leq_p$ monotone: for any 2 DBP's where $\mathcal{B} \geq_p \mathcal{B}'$: $\mathcal{D}^*_{\mathcal{T}}(\mathcal{B}) \geq_p \mathcal{D}^*_{\mathcal{T}}(\mathcal{B}')$.*

Proof. This follows directly from Definition 5.1.11 and the fact that the valuation $(\cdot)^{\mathcal{B},I}$ is $\leq_p$ monotone. □

Theorem 5.1.13. *$\mathcal{D}^*_{\mathcal{T}}$ is an approximator of $\mathcal{D}_{\mathcal{T}}$.*

Proof. By definition 2.6.1, we need to prove three things to show that $\mathcal{D}^*_{\mathcal{T}}$ is an approximator of $\mathcal{D}_{\mathcal{T}}$.

1. $\mathcal{D}^*_{\mathcal{T}}$ is $\leq_p$ monotone.
2. For each DPWS $\mathcal{Q}$, we have that $\mathcal{D}^c_{\mathcal{T}}((\mathcal{Q},\mathcal{Q})) \leq_p \mathcal{D}_{\mathcal{T}}(\mathcal{Q})$
3. For each DPWS $\mathcal{Q}$, we have that $\mathcal{D}^l_{\mathcal{T}}((\mathcal{Q},\mathcal{Q})) \geq_p \mathcal{D}_{\mathcal{T}}(\mathcal{Q})$

The first follows directly from Proposition 5.1.12. The second statement follows straightforward from the definition of $\mathcal{D}^*_{\mathcal{T}}$:

$$(\{I \mid (T_A)^{(\mathcal{Q},\mathcal{Q}),I} \neq \mathbf{f}\})_{A\in\mathcal{A}}$$

$$= (\{I \mid (T_A)^{\mathcal{Q},I} \neq \mathbf{f}\})_{A\in\mathcal{A}}$$

$$= (\{I \mid (T_A)^{\mathcal{Q},I} = \mathbf{t}\})_{A\in\mathcal{A}}$$

where we use Proposition 5.1.12 and the fact that the valuation $(\cdot)^{\mathcal{Q},I}$ is two-valued.

The proof for the third statement is similar. □

The stable operator $\mathcal{D}^{st}_{\mathcal{T}}$ is defined for dAEL as $\mathcal{D}^{st}_{\mathcal{T}}(\mathcal{Q}) = \text{lfp}(\mathcal{D}^*_{\mathcal{T}}(\cdot,\mathcal{Q})^c)$. Different fixpoints of these operators lead to different semantics as discussed in Section 2.5;

Definition 5.1.14. Let $\mathcal{T}$ be a distributed theory.

- A *supported model* of $\mathcal{T}$ is a fixpoint of $\mathcal{D}_{\mathcal{T}}$.
- The *Kripke-Kleene model* of $\mathcal{T}$ is the $\leq_p$-least fixpoint of $\mathcal{D}^*_{\mathcal{T}}$.
- A *partial stable model* of $\mathcal{T}$ is a DBP $\mathcal{B}$, such that $\mathcal{B}^c = \mathcal{D}^{st}_{\mathcal{T}}(\mathcal{B}^l)$ and $\mathcal{B}^l = \mathcal{D}^{st}_{\mathcal{T}}(\mathcal{B}^c)$.
- A *stable model* of $\mathcal{T}$ is a DPWS $\mathcal{Q}$, such that $(\mathcal{Q}, \mathcal{Q})$ is a partial stable model of $\mathcal{T}$.
- The *well-founded model* of $\mathcal{T}$ is the least precise partial stable model of $\mathcal{T}$.

To illustrate these operators, fixpoints and semantics we will make use of two examples. Using these, we will look into the different semantics and the intuitions behind them.

Example 5.1.15. Imagine a situation with two agents, a mother and a father that have a 6-year old child together: $A = \{M, D\}$. The child really wants candy and asks his parents. His father answers: "You can have a piece of candy if it is ok for your mom.". When he asks the mother, she answers: "You may take some candy, if your father says so.". In dAEL, with a vocabulary $\Sigma = \{c\}$ this can be modelled as $\mathcal{T} = (T_D, T_M)$, with:

$$T_D = \{K_M(c) \Rightarrow c\} \qquad\qquad T_M = \{K_D(c) \Rightarrow c\}.$$

Luckily, the child has an inherent comprehension of dAEL. To find out whether or not he will get candy, he studies which possible semantics the parents can choose. There exist four possible world structures for each agent [1]:

1. The empty possible world set or inconsistent belief: $\emptyset$, denoted as $\top$.
2. The belief of c: $\{\{c\}\}$
3. The disbelief of c: $\{\emptyset\}$
4. The lack of knowledge: $\{\emptyset, \{c\}\}$, denoted as $\bot$.

The semantic operator associated to this theory is:

$$\begin{aligned}\mathcal{D}_{\mathcal{T}}(\mathcal{Q}) &= (\mathcal{Q}'_D, \mathcal{Q}'_M) \\ &= (\{I \mid \neg(K_M(c))^{\mathcal{Q}} \vee c^I = \mathbf{t}\},\ \{I \mid \neg(K_D(c))^{\mathcal{Q}} \vee c^I = \mathbf{t}\})\end{aligned}$$

[1] The reason we use $\top$ for the smallest and $\bot$ for the largest set is that they are the largest respectively the smallest element according to the knowledge order.

And the approximator is $\mathcal{D}^*_{\mathcal{T}}(\mathcal{B}) = (\mathcal{D}^c_{\mathcal{T}}(\mathcal{B}), \mathcal{D}^l_{\mathcal{T}}(\mathcal{B}))$, where:

$$\mathcal{D}^c_{\mathcal{T}}(\mathcal{B}) = (\{I \mid \neg(K_M(c))^{\mathcal{B}} \vee c^I \neq \mathbf{f}\}, \{I \mid \neg(K_D(c))^{\mathcal{B}} \vee c^I \neq \mathbf{f}\})$$

$$\mathcal{D}^l_{\mathcal{T}}(\mathcal{B}) = (\{I \mid \neg(K_M(c))^{\mathcal{B}} \vee c^I = \mathbf{t}\}, \{I \mid \neg(K_D(c))^{\mathcal{B}} \vee c^I = \mathbf{t}\})$$

Supported models are fixpoints of $\mathcal{D}_{\mathcal{T}}$. There are two *supported models*, namely $(\bot_D, \bot_M)$ and $(\{\{c\}\}_D, \{\{c\}\}_M)$: either both Dad and Mom agree to giving candy or none of them does.

The Kripke-Kleene fixpoint is the least fixpoint of $\mathcal{D}^*_{\mathcal{T}}$ with the precision relation. The *Kripke-Kleene model* is $((\bot, \{\{c\}\})_D, (\bot, \{\{c\}\})_M)$. So in the Kripke-Kleene semantics it is unknown for both Dad and Mom whether the child can have candy. However, from none of their theories it follows that the child can have no candy.

The DPWS $\bot := (\bot_D, \bot_M)$ is the (unique) *stable model*: $(\bot, \bot)$ is a *partial stable* model, since $\bot_D = \text{lfp}(\mathcal{D}^*_{\mathcal{T}}(\cdot, \bot))^c$ and $\bot_M = \text{lfp}(\mathcal{D}^*_{\mathcal{T}}(\cdot, \bot))^l$. As such, $(\bot, \bot)$ is the *well-founded model.*

The stable and well-founded semantics only derive knowledge that is "grounded" in the theory: knowledge is only derived if there is a non-self supporting reason. This is a reasonable way of deriving knowledge from the theories. Differences between the stable and well-founded semantics occur when loops over negation occur as next example shows:

Example 5.1.16. Assume the same situation as in the example above, but with divorced parents, a mother and a father that have a 6-year old child together: $A = \{M, D\}$. The child really wants candy and asks his parents. His father answers: "You get a piece of candy from me, if your mom doesn't approve.". When he asks the mother, she answers: "If your father says no, you may take some candy.". These statements can be modelled in dAEL as:

$$T_D = \{\neg K_M(c) \Rightarrow c\} \qquad\qquad T_M = \{\neg K_D(c) \Rightarrow c\}.$$

This theory has no supported models, since no model is "consistent" with this theory. The Kripke-Kleene model is again $((\bot, \{\{c\}\})_D, (\bot, \{\{c\}\})_M)$. However, this theory has two stable models: $\bot := (\bot_D, c_M)$ and $(c_D, \bot_M)$. The well-founded model is $((\bot, \{\{c\}\})_D, (\bot, \{\{c\}\})_M)$, the non-exact well-founded model is due to the loop over negation in this theory.

5.2 A Mapping from dAEL to AEL

We now describe a mapping from dAEL to AEL. We prove that for distributed theories that do not contain any inconsistency, the semantics for dAEL match the corresponding semantics for AEL. In the case of partially inconsistent distributed theories, the semantics do not coincide: dAEL allows for a single agent to have inconsistent beliefs, whereas AEL has no mechanism to encapsulate an inconsistency in a similar way. This capacity of dAEL to encapsulate inconsistencies is a desirable feature, for instance for the application of dAEL in access control, where it facilitates, for example, to *isolate* a faulty agent, as we will show below.

It has been noted before, by Vennekens et al. [2007] and Vlaeminck et al. [2012] that natural embeddings of certain "stratified" theories in AEL fail when there is the possibility of inconsistent knowledge. They have presented the notion of *permaconsistent* theories as a criterion for their embeddings to work. In this section, we show

- how to generalise permaconsistency to dAEL,
- that for permaconsistent theories, the mapping we define indeed preserves semantics, and
- that a weaker criterion (being universally consistent) works for three of the five semantics.

We first generalize the notion of permaconsistency [Vlaeminck et al., 2012] to the distributed case:

Definition 5.2.1. A distributed theory $\mathcal{T}$ is *permaconsistent* if for each $A \in \mathcal{A}$ and each theory T' that can be constructed from $(\mathcal{T})_A$ by replacing any occurrence of formulas $K_B\varphi$ not nested under a modal operator by $\mathbf{t}$ or $\mathbf{f}$ independently of each other is consistent.

Note that with "replacing all occurences" it is meant that every occurence can be independently and arbitrarily replaced by $\mathbf{t}$ or $\mathbf{f}$.

5.2.1 A translation for vocabularies, theories and structures

Intuitively, the mapping from dAEL to AEL adds an argument to each symbol of the vocabulary containing an agent, the agent from whose view this symbol is interpreted. In Example 5.1.15, c would be transformed to a predicate $c/1$, where $c(M)$ and $c(D)$ state whether the mother respectively the father gives candy.

The mapping from dAEL to AEL consists of a collection of translation functions. These functions translate syntactic constructs of dAEL like formulas and distributed theories and semantic constructs of dAEL like DPWS's and distributed belief pairs into the corresponding constructs of AEL. We denote each of these translation functions by τ with some subscript, where the subscript indicates the type of the output of the translation function. For example, we use τ_T to denote the translation function from distributed theories to AEL-theories, because T is the usual symbol used for an AEL-theory.

Given a vocabulary Σ of a distributed theory $\mathcal{T}$ in dAEL, the AEL translation of $\mathcal{T}$ will be written in a slightly modified vocabulary Σ' that adds the extra argument to the predicates and functions.

Definition 5.2.2. Given a vocabulary Σ, we define Σ' to be the vocabulary consisting of the same predicate and function symbols as Σ but with the arity of each predicate and function symbol that is not a domain element increased by one.

The additional argument of each predicate and function symbol refers to the agent whose beliefs about the predicate/function symbol we are using to interpret the symbol. Given an n-ary function symbol $f \in \Sigma$, we will therefore interpret f as an $n{+}1$-ary function symbol in AEL, where $f(a_1, \dots, a_n, a_{n+1})$ should be interpreted as the interpretation of $f(a_1, \dots, a_n)$ according to agent a_{n+1}. Since we assume all functions to be total, $f(a_1, \dots, a_n, a_{n+1})$ also needs to be interpreted when a_{n+1} is not an agent. Since it does not matter which value we give to $f(a_1, \dots, a_n, a_{n+1})$ (this will follow from our particular translation) in this case, we fix an arbitrary element δ in our domain D to assign to such defective terms.

We use the following notational conventions in this section: ϕ denotes a dAEL formula over Σ , φ denotes an AEL formula over Σ', I denotes a Σ-structure, and J denotes a Σ'-structure. We also use $\mathcal{L}_d$ to denote the language of dAEL and $\mathcal{L}_k$ to denote the language of AEL. With a superscript $\mathcal{L}^{\Sigma}$ to a language $\mathcal{L}$, it is explicitated that $\mathcal{L}$ is a language over vocabulary Σ.

Definition 5.2.3. Given a Σ-term t and an agent A, we define the Σ'-term t_A recursively as follows:

- $x_A := x$ for each variable x
- $d_A := d$ for each domain element $d \in D$.

- $(f(t_1, \ldots, t_n))_A := f(t_{1A}, \ldots, t_{nA}, A)$

Definition 5.2.4. We define the function $\tau_\varphi : \mathcal{A} \times \mathcal{L}_d^\Sigma \to {\mathcal{L}_k}^{\Sigma'}$ as follows:

- $\tau_\varphi(A, P(t_1, \ldots, t_n)) := P(t_{1A}, \ldots, t_{nA}, A)$
- $\tau_\varphi(A, \neg\phi) = \neg\tau_\varphi(A, \phi)$
- $\tau_\varphi(A, \phi \wedge \psi) = \tau_\varphi(A, \phi) \wedge \tau_\varphi(A, \psi)$
- $\tau_\varphi(A, (\forall x)\phi) = (\forall x)\tau_\varphi(A, \phi)$
- $\tau_\varphi(A, K_B\phi) = K\tau_\varphi(B, \phi)$

Definition 5.2.5. For a distributed theory $\mathcal{T}$, define $\tau_T(\mathcal{T}) := \bigcup_{A\in\mathcal{A}} \tau_\varphi(A, T_A)$.

To give some intuition, we now show how the mapping for theories works in Example 5.1.15.

Example 5.2.6. Given a set of agents $\mathcal{A} = \{M, D\}$ and the distributed theory $\mathcal{T} = (\{K_D c \Rightarrow c\}, \{K_M c \Rightarrow c\})$, we calculate:

$$\begin{aligned}
\tau_T(\mathcal{T}) =& \tau_T(\{K_D c \Rightarrow c\}, \{K_M c \Rightarrow c\}) \\
&= \{\tau_\varphi(M, K_D c \Rightarrow c)\} \cup \{\tau_\varphi(D, K_M c \Rightarrow c)\} \\
&= \{\tau_\varphi(M, \neg K_D c \vee c),\ \tau_\varphi(D, \neg K_M c \vee c)\} \\
&= \{\neg\tau_\varphi(M, K_D c) \vee \tau_\varphi(M, c),\ \neg\tau_\varphi(D, K_M c) \vee \tau_\varphi(D, c)\} \\
&= \{\neg\tau_\varphi(D, c) \vee c(M),\ \neg\tau_\varphi(M, c) \vee c(D)\} \\
&= \{\neg c(D) \vee c(M),\ \neg c(M) \vee c(D)\} \\
&= \{c(D) \Rightarrow c(M),\ c(M) \Rightarrow c(D)\}
\end{aligned}$$

Example 5.2.7. To illustrate how this mapping isolates faulty agents, assume following distributed theory $\mathcal{T}$ that is not permaconsistent:

$$\mathcal{T} = (\{c \wedge \neg c\}, \{c\})$$

In this case:

$$\begin{aligned}\tau_T(\mathcal{T}) =& \tau_T(\{c \wedge \neg c\}, \{c\})\\ &= \{\tau_\varphi(c \wedge \neg c)\} \cup \{\tau_\varphi(c)\}\\ &= \{\tau_\varphi(M, c \wedge \neg c)\} \cup \{\tau_\varphi(D, c)\}\\ &= \{c(M) \wedge \neg c(M), c(D)\}\\ &= \{\mathbf{f}\}\end{aligned}$$

Assume a third agent with policy $\{K_D(c) \implies c\}$, it is clear that in $\mathcal{T}$ this agent would still be able to decide to give candy to the child, while this is not the case in $\tau_T(\mathcal{T})$.

We now define a transformation for the semantical structures in dAEL. To map a DPWS $\mathcal{Q}$ to a PWS Q, we define a mapping

$$\tau_Q(\mathcal{Q}) = Q$$

such that there is a structure $J \in Q$ for each indexed family in $\{(I_A)_{A\in\mathcal{A}} | I_A \in Q_A\}$ (intuitively, every tuple such that there is one structure for each agent, picked from the relevant PWS Q_A).

Definition 5.2.8. For an indexed family $\mathcal{I} = (I_A)_{A\in\mathcal{A}}$ of Σ-structures, we define the Σ'-structure $\tau_J(\mathcal{I})$ by defining

$$f^{\tau_J(\mathcal{I})}(d_1, \ldots, d_n, A) := f^{I_A}(d_1, \ldots, d_n)$$

for each n-ary function symbol $f \in \Sigma$ and all $d_1, \ldots, d_n \in D$, and by defining

$$P^{\tau_J(\mathcal{I})}(d_1, \ldots, d_n, A) \text{ iff } P^{I_A}(d_1, \ldots, d_n)$$

for each n-ary predicate symbol $P \in \Sigma$ and $d_1, \ldots, d_n \in D$.

Definition 5.2.9. For a DPWS $\mathcal{Q}$, we define $\tau_Q(\mathcal{Q}) := \{\tau_J((I_A)_{A\in\mathcal{A}}) \mid I_A \in \mathcal{Q}_A$ for every $A \in \mathcal{A}\}$.

Definition 5.2.10. For a distributed belief pair $\mathcal{B}$, define

$$\tau_B(\mathcal{B}) := (\tau_Q(\mathcal{B}^c), \tau_Q(\mathcal{B}^l))$$

We show this again using Example 5.1.15.

Example 5.2.11. A possible DPWS for this example is one where the father knows to give candy, while the mother does not know anything:

$$\mathcal{Q} = (\{\{\}, \{c\}\}_M, \{\{c\}\}_D))$$

We calculate

$$\begin{aligned}\tau_Q(\mathcal{Q}) &= \tau_Q(\{\{\}, \{c\}\}_M, \{\{c\}\}_D) \\ &= \{\tau_J(\{\}_M, \{c\}_D), \tau_J(\{c\}_M, \{c\}_D) \\ &= \{\{c(D)\}, \{c(M), c(D)\}\}\end{aligned}$$

5.2.2 Proof of correctness

The main result of this section is that the above mapping from dAEL to AEL preserves all semantics in case $\mathcal{T}$ is permaconsistent.

Theorem 5.2.12. *Let $\sigma \in \{\mathsf{Sup}, \mathsf{KK}, \mathsf{PSt}, \mathsf{St}, \mathsf{WF}\}$ be a semantics[a], let $\mathcal{T}$ be a permaconsistent distributed theory, and let $\mathcal{Q}$ be a distributed possible world structure and $\mathcal{B}$ be a distributed belief pair. Then $\mathcal{B}$ or $\mathcal{Q}$ (depending on the semantics) is a σ-model of $\mathcal{T}$ iff $\tau_B(\mathcal{B})$, respectively $\tau_Q(\mathcal{Q})$ is a σ-model of $\tau_T(\mathcal{T})$.*

[a]We use Sup, KK, PSt, St, WF as shorthands for respectively the Supported, Kripke-Kleene, Partial Stable, Stable, and Well-founded semantics.

We also present a weaker criterion that preserves models for three out of the five semantics.

Definition 5.2.13. We call a DPWS $\mathcal{Q}$ *universally consistent* if $(\mathcal{Q})_A \neq \emptyset$ for all $A \in \mathcal{A}$.

Definition 5.2.14. We call a distributed belief pair $\mathcal{B}$ *universally consistent* if $\mathcal{B}^l$ is universally consistent.

If $\mathcal{B}$ is universally consistent, so is $\mathcal{B}^c$.

Definition 5.2.15. Let $\sigma \in \{\mathsf{Sup}, \mathsf{KK}, \mathsf{PSt}, \mathsf{St}, \mathsf{WF}\}$ be a semantics. We call a distributed theory $\mathcal{T}$ *universally consistent under σ* iff every σ-model of $\mathcal{T}$ is universally consistent.

The following theorem states that the mapping from dAEL to AEL is faithful for universally consistent models of a distributed theory for three out of the five semantics.

Theorem 5.2.16. *Let $\sigma \in \{\mathsf{Sup}, \mathsf{St}\}$ be a semantics, let $\mathcal{T}$ be a distributed theory, let $\mathcal{Q}$ be a universally consistent DPWS. Then $\mathcal{Q}$ is a σ-model of $\mathcal{T}$ iff $\tau_Q(\mathcal{Q})$ is a σ-model of $\tau_T(\mathcal{T})$. Let $\mathcal{B}$ be a universally consistent distributed belief pair, then $\mathcal{B}$ is a PSt-model of $\mathcal{T}$ iff $\tau_B(\mathcal{B})$ is a PSt-model of $\tau_T(\mathcal{T})$.*

Since permaconsistent implies universal consistency (Theorem 5.2.17), Theorem 5.2.16 implies 5.2.12 for supported and (partial) stable semantics.

Theorem 5.2.17. *Let $\sigma \in \{\mathsf{Sup}, \mathsf{KK}, \mathsf{PSt}, \mathsf{St}, \mathsf{WF}\}$. If $\mathcal{T}$ is permaconsistent, then $\mathcal{T}$ is universally consistent under σ.*

Proof. Suppose $\mathcal{T}$ is permaconsistent. Now, for each agent $A \in \mathcal{A}$ and each theory T' that can be constructed from $T_A = (\mathcal{T})_A$ by replacing any non-nested occurrence of modal literals by $\mathbf{t}$ or $\mathbf{f}$ is consistent. Now, for each agent A, let T'_A the theory constructed from T_A by replacing all non-nested occurrences of modal literals by $\mathbf{t}$ if they occur in a negative context (under an odd number of negations) and by $\mathbf{f}$ otherwise. This theory is clearly stronger than T_A. Since $\mathcal{T}$ is permaconsistent, T'_A is satisfiable, so let I_A be a model of T'_A. In this case, it holds that $T_A^{(\bot,\top),I_A} = \mathbf{t}$ (since T_A is weaker than T'_A).

From this, we find that for each agent A, $\{I \mid T_A^{(\bot,\top),I}\}$ is non-empty and thus that $\mathcal{D}_{\mathcal{T}}^{*\,*}(\bot,\top)$ is universally consistent. From the definitions in AFT it follows directly that each model of $\mathcal{T}$ (under any of the semantics), is more precise than $\mathcal{D}_{\mathcal{T}}^{*}(\bot,\top)$. Furthermore, if $\mathcal{B}' \geq_p \mathcal{B}$ and $\mathcal{B}$ is universally consistent, then so is $\mathcal{B}'$. $\square$

Example 5.2.18. The reverse of Theorem 5.2.17 does not hold as can be seen for example by a theory $\{p \Rightarrow K_A p, K_A p \Rightarrow p\}$ with one agent A.

Since the proofs of these theorems and intermediate results are very technical and do not bring much insight, we omit them here. They can be found in Appendix A.1.

5.3 Extension: Complex terms for agents: $dAEL^+$

When a user specifies his access control policy, he might want to refer to the knowledge of an agent, even when he does not know his identity. For example, assume there is a function $Owner : file \rightarrow Agents$, that maps files to the agent responsible for managing that file. The interpretation of that function can differ from agent to agent, and even within the different possible worlds in an agent's PWS. This allows an agent to make policies like :

$$\forall x\ p : K_{Owner(x)} Access(p, x) \Leftrightarrow Access(p, x).$$

And while the different possible worlds of that agent can have different interpretations for $Owner$, the agent will allow access if in all of his possible worlds the owner allows access.

We define syntax and valuation as adjustments to Definition 5.1.1, 5.1.5 and 5.1.8. Given these modification of the valuations, the operator and approximator and by consequence the different semantics are defined exactly as before, so we will not repeat that here.

Definition 5.3.1. We define the language $dAEL^+$ of distributed autoepistemic logic with terms using the standard recursive rules of first-order logic, augmented with:

$$K_t(\psi) \text{ is a formula if } \psi \text{ is a formula and } t \in \mathbb{T}$$

The intuitive reading of $K_t(\psi)$ is "t is an agent and t knows ψ". So if the term t does not denote an agent, $K_t(\psi)$ will be interpreted to be false.

We assume that in $dAEL^+$ the set of agents is part of the domain ($\mathcal{A} \subseteq D$) and that $dAEL^+$ contains a dedicated unary predicate Ag, whose interpretation is assumed to always be $\mathcal{A}$.

Definition 5.3.2. The *value* of a $dAEL^+$ sentence φ in $\mathcal{Q}, I$ (denoted $\varphi^{\mathcal{Q},I}$) is defined inductively by the standard recursive rules for first-order logic, augmented with:

$$(K_t\varphi)^{\mathcal{Q},I} = \mathbf{t} \qquad \text{if } t^I \in \mathcal{A} \text{ and } \varphi^{\mathcal{Q},J} = \mathbf{t} \text{ for each } J \in (\mathcal{Q})_{t^I}$$

Definition 5.3.3. The *value* of a $dAEL^+$ sentence φ with respect to DBP $\mathcal{B}$ and interpretation I (notation $\varphi^{\mathcal{B},I}$) is defined inductively by replacing the fifth rule in the recursive definition of the three-valued valuation of an AEL formula (Definition 2.6.3) by:

$$(K_A\varphi)^{\mathcal{B},I} = \begin{cases} \mathbf{t} & \text{if } t^I \in Ag^I \text{ and } \varphi^{\mathcal{B},I'} = \mathbf{t} \text{ for all } I' \in (\mathcal{B}^c)_{t^I} \\ \mathbf{f} & \text{if } t^I \notin Ag^I \text{ or } \varphi^{\mathcal{B},I'} = \mathbf{f} \text{ for some } I' \in (\mathcal{B}^l)_{t^I} \\ \mathbf{u} & \text{otherwise} \end{cases}$$

5.4 Applying dAEL to Access Control

An *access control policy* is a set of norms defining which principal is to be granted access to which resource under which circumstances. Specialized logics called *access control logics* were developed for representing policies and access requests and reasoning about them. To make a connection between the logic and the access control, most logic based-approaches grant an access request if and only if it is logically entailed by the policy. Another approach might be to only disallow access if the negation of a request is entailed, which is a weaker form of access control, since then access is given if both the request and the negation of the request are not entailed by the policy. We will follow the first approach, for clear security reasons. However, generalising the application of our logic to the second approach or to other approaches (approaches in between) is straightforward.

There is a large variety of access control logics. Many of them use a modality $k\,\mathit{says}$ indexed by a principal k [Abadi et al., 1993, Genovese, 2012]. *says*-based access control logics are designed for systems in which different principals can issue statements that become part of the access control policy. $k\,\mathit{says}\,\varphi$ is usually explained informally to mean that k supports φ [Abadi, 2008, Garg and Pfenning, 2012, Genovese, 2012]. This means that k has issued statements that – together with additional information present in the system – imply φ. Different access control logics vary in their account of which rules of inference and which additional information may be used in deriving statements that k supports from the statements that k has explicitly issued.

We illustrate the *says*-modality in access control by showing how it is employed to delegate authority. Suppose that principal A has control over a resource r, i.e., that any principal i has access to r if and only if A says that i has access to r. Now A can delegate to principal C the decision whether B has access to r by issuing the statement

$$(C\,\mathit{says}\,\mathit{access}(B, r)) \Rightarrow \mathit{access}(B, r). \tag{5.1}$$

If C issues $\mathit{access}(B, r)$, then (5.1) implies $\mathit{access}(B, r)$, i.e., $A\,\mathit{says}\,\mathit{access}(B, r)$, so B has access to r.

Note that we used the fact that C said $\mathit{access}(B, r)$ in order to derive what A supports from what A explicitly said: we assumed $A\,\mathit{says}\,(C\,\mathit{says}\,\mathit{access}(B, r))$ based on $C\,\mathit{says}\,\mathit{access}(B, r)$. To make delegation work in general, practically all *says*-based logics statements allow us to derive $j\,\mathit{says}\,(k\,\mathit{says}\,\varphi)$ from $k\,\mathit{says}\,\varphi$. Note that in epistemic terminology, by identifying $k\,\mathit{says}$ with K_k, this can be called mutual positive introspection between principals.

Many state-of-the-art access control logics are based on intuitionistic rather than classical logic. Garg [2009] justifies the use of intuitionistic logic in access control on the basis of the security principle that when access is granted to a principal k, it should be known where k's authority comes from. For example BL, an access control logic with support for system state and explicit time [Garg, 2009, Garg and Pfenning, 2012], is an intuitionistic modal logic with support for mutual positive introspection but not for mutual negative introspection between principals. dAEL, on the other hand, is based on classical logic, and supports mutual negative introspection between principals. In order to justify our claim that dAEL is a good access control logic, we discuss these two differences between BL and dAEL.

We illustrate the advantage of mutual negative introspection by showing how it allows to correctly handle statements whose goal it is to deny or revoke access rights. Suppose A is a professor with control over a resource r, B is a PhD student of A who needs access to r, and C is a postdoc of A supervising B. A wants to grant B access to r, but wants to grant C the right to deny B's access to r. A natural way for A to do this is to issue the statement $(\neg C\, \mathit{says}\, \neg \mathit{access}(B, r)) \Rightarrow \mathit{access}(B, r)$. This should have the effect that B has access to r unless C denies him access. However, this effect can only be achieved if we assume the *says*-modality to allow mutual negative introspection: A must know that C does *not* issue a statement $\neg \mathit{access}(B, r)$ to derive that B has access rights.

In order to derive statements of the form $\neg k\, \mathit{says}\, \varphi$, we have to assume the statements issued to be a complete characterization of what the agent supports, like the “All I Know” assumption for AEL [Levesque, 1990]. Together with support for mutual positive and negative introspection, this motivates the use of dAEL as a viable access control logic.

The use of dAEL to model the *says*-modality can also be justified directly from the intuitive reading of the *says*-modality mentioned above if we assume that the additional information that may be used for deriving which statements a principal j supports from the statements j has issued consists precisely of all information of the form $k\, \mathit{says}\, \varphi$ or $\neg(k\, \mathit{says}\, \varphi)$. We consider this a reasonable assumption, given that the information of this form is the only information that can be considered independent of the point of view of any principal in a distributed system.

When using a policy to decide who gets access, there is a key security principle that has to be guaranteed: "When access is granted to a principal k, it should be known where k's authority comes from” [Garg, 2009]. This points to the responsibility the agents have in managing their resources, and the accountability such a system needs when something goes wrong. We now argue how this principle motivates us to use the well-founded semantics, using two examples where access can be delegated and revoked between agents.

5.4.1 Examples: Delegation and Revocation of Access

When principals delegate access rights to others, delegation chains can be formed. There are different ways to treat these delegation chains when revoking rights, which give rise to different revocation schemes [Hagström et al., 2001, Cramer et al., 2015]. Of these revocation schemes, the one with the strongest effect is called the Strong Global Negative (SGN) revocation scheme: In this scheme, revocation is performed by issuing a negative authorization which dominates over positive authorizations and whose effect propagates forward. We will show that our dAEL model of SGN revocation behaves precisely like it was defined by Cramer et al. [2015].

Assume a setting with four agents $\mathcal{A} = \{A, B, C, D\}$ and a resource r. Suppose that A controls r and that A wants to delegate access right to other principals, whom A wants to have the power to revoke rights from other users according to the SGN revocation scheme. A principal k can delegate access right to a principal j for all resources it has access to by issuing the statement $\mathit{deleg\_to}(j)$, and can revoke access right from j by issuing the statement $\mathit{revoke}(j)$. Assuming that access will be granted to a principal k iff $A\ \mathit{says}\ \mathit{access}(k, r)$, A can ensure that the statements of the form $\mathit{deleg\_to}(j)$ and $\mathit{revoke}(j)$ will be interpreted as delegation and SGN revocation by putting following constraint on the predicate access:

$$\begin{aligned}
&\mathit{access}(A, r).\\
&\mathit{access}(j, r) \Leftarrow\\
&\quad \exists k\ (K_A \mathit{access}(k, r) \wedge K_k \mathit{deleg\_to}(j)) \wedge\\
&\quad \neg\exists i\ (K_A \mathit{access}(i, r) \wedge K_i \mathit{revoke}(j)).
\end{aligned}$$

This defines who has access to what: all principals k have access to r if $K_A \mathit{Access}(k, r)$. It starts with a base case stating that agent A (who we assume to be the owner) has access. It uses this to recursively state that all principals that get access from A, or from someone who has access to r, get access to r unless they are explicitly distrusted by A or one of his trustees.

Given a certain semantics for dAEL, we will follow the standard in access control to grant k access to r only if $K_A \mathit{access}(k, r)$ holds in all models. We do this since if there is a model (a possible situation) where someone does not have access, there is reason to assume that he should not be given access, so for security reasons we will not give him access. With this interpretation of the semantics, the partial stable semantics and the well-founded semantics coincide, since the well-founded model is the least precise partial stable model. Therefore, we ignore the partial stable semantics for the discussion of semantics in this section.

We now argue using two example scenarios why we believe the well-founded semantics to be the best choice when applying dAEL to access control. In both scenarios, we assume that A controls r and that A has the above constraints on access, allowing them to perform SGN revocation.

Example 5.4.1 (Scenario 1)**.** In the first scenario, we suppose A has issued the statements $\mathit{deleg\_to}(B)$ and $\mathit{deleg\_to}(C)$, that B has issued the statements $\mathit{revoke}(C)$ and $\mathit{deleg\_to}(D)$, and that C has issued the statements $\mathit{revoke}(B)$ and $\mathit{deleg\_to}(D)$. We visualize this scenario in Figure 5.1.

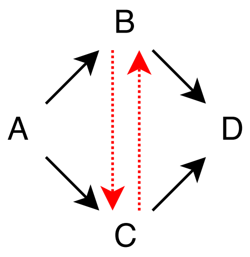

Figure 5.1: Delegation-revocation example 1. Delegation is marked by a full arrow, revocation by a (red) dotted arrow.

By issuing $\mathit{revoke}(C)$, B is attempting to revoke C's access right (and vice versa). Of course, this attempt is only successful if B has access. So C should have access right iff B does not. Since the scenario is symmetric between B and C, and we stated above that agent B can only have access if $\mathit{Access}(B, r)$ is true in all models, both agents should be denied access right. However, D does have access in this situation, violating the security principle that it should be clear who is accountable for giving D access. The scenario contains a conflict that cannot be automatically resolved. At this point, A as the principal with control over r will have to manually resolve the conflict by removing access from at least one of them (B or C). In practice, it may take A some time to study the situation and perform this manual resolution. During this time, the system should still respond to access requests. It is clear that in this scenario conflict resolution is in order. There exists different ways to handle conflicts, see [Jajodia et al., 2001]. Important for the choice of semantics is that it points out conflicts.

Consider the statements issued by the principals as a distributed theory in dAEL. This theory has different models depending on the choice of semantics. We present the models by presenting a set of expressions X^t where X is a principle and t the truth value of $K_A\, access(X, r)$ in the model. There are two supported models $\{A^{\mathbf{t}}, B^{\mathbf{t}}, C^{\mathbf{f}}, D^{\mathbf{t}}\}$ and $\{A^{\mathbf{t}}, B^{\mathbf{f}}, C^{\mathbf{t}}, D^{\mathbf{t}}\}$. These are also the stable models. So, according to the supported and stable semantics: A and D have access, while B and C do not. The Kripke-Kleene model and the well-founded model are identical: $\{A^{\mathbf{t}}, B^{\mathbf{u}}, C^{\mathbf{u}}, D^{\mathbf{u}}\}$. This model is not exact: The truth-value of the statements $A\, says\, access(X, r)$, with $X \in \{B, C, D\}$ is unknown. Only A is given access according to the Kripke-Kleene and well-founded semantics. Note that all supported and stable models both grant access to D. Given our above argument against granting access to D, these semantics cannot be considered viable for this application. The Kripke-Kleene and well-founded model of this theory gives access precisely to the principal that should have access according to our above discussion. Thus these semantics, while not based on intuitionistic logic, are faithful to the motivation that Garg and Pfenning [2012] gave for using intuitionistic logic in access control. Furthermore, they stress the existing conflict between B and C by making their access right status undefined.

Example 5.4.2 (Scenario 2)**.** Consider a second scenario, in which A has issued the statement *deleg_to*(B) and C has issued the statements *deleg_to*(C) and *revoke*(B). We visualize this scenario in Figure 5.2.

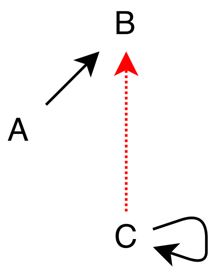

Figure 5.2: Delegation-revocation example 2: Delegation is marked by a full arrow, revocation by a (red) dotted arrow

Here C should clearly not have access, because the only principal granting her access is C herself. Hence C's revocation of B's access right does not have any effect, so B should be granted access. The Kripke-Kleene model $\{A^{\mathbf{t}}, B^{\mathbf{u}}, C^{\mathbf{u}}, D^{\mathbf{f}}\}$ of the distributed theory corresponding to this scenario is not exact; it is unknown whether B and C have access, which clearly diverges from our requirements. The well-founded model $\{A^{\mathbf{t}}, B^{\mathbf{t}}, C^{\mathbf{f}}, D^{\mathbf{f}}\}$ on the other hand correctly computes this desired outcome.

These scenarios illustrate our motivation of using the well-founded semantics for access-control applications.

These findings are in line with the findings of Denecker et al. [2011], who strongly argued in favour of the well-founded semantics in AEL.

5.5 Conclusion

Motivated by an application in access control, we extended AEL to a distributed setting, resulting in *dAEL: distributed autoepistemic logic.* dAEL allows for a set of agents to each have their own theory and refer to each others knowledge. For this, the K operator of AEL is replaced by an indexed operator K_A, where A refers to an agent. In a setting where a set of agents communicate openly and honestly, we needed a logic where this open communication was inherent. Assuming the theories of the different agents to be characterizations of their public commitments, we made dAEL a logic with full (positive and negative) mutual introspection. This means that every agent can refer to the knowledge (or equivalently in our framework: the consequences of the commitments) of other agents and the lack thereof.

We defined the semantics of this logic using AFT and generalized 5 different semantics from the single agent setting to dAEL: Supported semantics, Kripke-Kleene semantics, (Partial) Stable semantics and Well-founded semantics. Other semantics with other advantages have been studied in recent work [Bogaerts, 2015], and can also be generalised to dAEL, using the AFT framework.

We studied its relation to standard AEL and formulated an equivalence preserving mapping from dAEL to AEL for a subset of the logic. We showed that the embedding of dAEL in AEL is not always equivalence preserving and pinpointed the exact situations in which equivalence can be lost. Furthermore, we argued that in those situations, the semantics of dAEL is preferred over AEL's semantics, at least in the context of our applications. Further motivated by applications in access control, we defined an extension: $dAEL^+$: The extension of dAEL, where one can use functions and terms in general in the subscript of the K operator, allowing for statements like $K_{Owner(file)}Access(A, file)$, making a richer logic for our applications. We discussed the usability of this new logic in access control and illustrated it with examples.

Possible future research directions are the generalisation of other semantics for AEL to the distributed case, as discussed above, and the study of a decision procedure for dAEL. In this chapter we study the semantics of dAEL, but for practical applications, a decision procedure for dAEL or an expressively rich subset of it needs to be developed. The complexity of determining access rights based on a theory written in dAEL should be studied. Furthermore, the relation of dAEL to various existing access control models needs to be studied further.

6 A Possible-world Semantics for a multi-agent modal logic

When defining a semantics in the context of a multi-agent setting, a mathematical structure (or semantical structure) needs to be defined that allows to talk about the objective state of affairs but also allows to account for the knowledge that the agents have about the world. First-order logic has a mathematical formalism to talk about the objective state of affairs. The main question in modal logics is to find a mathematical way to describe the epistemic state of the agents. The key idea behind possible world semantics is that an epistemic state can be represented by a collection of possible worlds. This means that a world consists of

- An objective state of affairs
- For each agent a collection of worlds

The problem, however, is that with this definition, the notion of worlds is unfounded. The set of worlds does not exists, or cannot be constructed in standard set-theory. To show this, assume that this set of worlds exists, and take a world w where the belief of an agent A consists of all worlds w' that satisfy the constraint that w' is not a possible world of A in w' (or stated differently: the belief of A in w consists of all worlds in which A does not believe the objective state to be possible). In that case, w must satisfy one of the following 2 statements:

- A believes that w is possible. In that case, w is a world that satisfies the constraint that w is not a possible world of A in w, or equivalently: A does not believe that w is possible.
- A believes that w is not possible. In that case, w is a world that does not satisfy the constraint that w is not a possible world of A in w, or equivalently: A does believe that w is possible.

This shows that if the set of worlds where the epistemic state of an agent is a set of worlds would exist, this paradox exists.

In modal logics, this problem is avoided by using Kripke structures. When using Kripke structures, the set of potential worlds is fixed in advance and the logic is valuated relative to this class of worlds, by defining an access relation for each agent between the worlds and an agent knows something in a world w if it is true in all worlds accessible from w. This approach works well for many applications, but when studying only knowing, it no longer suffices. When evaluating only knowing, we want to talk about all worlds satisfying a certain formula, so we cannot afford to only talk about all worlds that are in the set of chosen worlds that satisfy that formula.

This same motivation is found when Belle et al. defined $\mathcal{COL}_m$ in [Belle and Lakemeyer, 2015]: a logic that studied the interaction between common knowledge ($\mathcal{C}$) and only knowing ($\mathcal{O}$). Based on work done by Fagin et al. [Fagin et al., 1991] they used a semantical method to create a sufficiently rich class of worlds such that this set of worlds is large enough to evaluate all formulas in $\mathcal{COL}_m$. Motivated by the large complexity in the work by Fagin et al. they propose a more simple method to define worlds. While Fagin et al. proposed a transfinite induction to a depth of ω^2, Belle et al. simplified this to a construction by induction to a depth of ω.

In the research presented in this thesis, we discovered anomalies that were present in [Belle and Lakemeyer, 2015]. We discovered a formula that should be satisfiable but cannot be made true given the semantical structure they proposed.

The formula stating that the only thing that is known is that something is not common knowledge cannot be true in any world they defined. However, it is clear that this is not a statement that could never be true.

In this chapter we will propose a new semantical structure for $\mathcal{COL}_m$, based on the work in [Fagin et al., 1991]. We propose a simplification of their k-worlds and knowledge assignments and will explore the only knowing modality in this framework. We present a semantical approach of defining μ-worlds (or worlds of depth μ) for every ordinal μ and make the link with Kripke structures for every λ-world, with λ a limit oridinal. We show that the set of satisfiable formulas increases when the depth of the world increases and prove that ω^2 worlds are deep enough to interpret every formula in $\mathcal{COL}_m$ not containing the only knowing operator. This is the same result as in [Fagin et al., 1991]. We then explore the only knowing operator O in this setting.

We will introduce the logic in steps, to improve readability. First we will introduce a logic $\mathcal{CL}_m$: a multi-agent modal logic with common knowledge, that uses a new semantical structure. Contrary to previous approaches (and to the approach in the previous chapter), this logic has very few assumptions. It does not assume introspection between agents or even introspection of an agent in its own knowledge and neither knowledge or belief to be truthful. However, given the set of semantical structures that will be defined below, subsets will be defined that do have these assumptions. The goal of this approach is not to develop a logic specifically for applications without for example introspection, but to allow to in fact "turn on or off" the needed modal assumptions, when using the logic. We will show the link between this semantical structure and the one introduced in [Belle and Lakemeyer, 2015]. The issues with Belle and Lakemeyer's logic are discussed and we propose a new semantics structure and a three-valued semantics that does not have these issues. The current proposal for a valuation is three-valued and for each formula there exists a semantical structure such that the valuation is two-valued for this formula in this structure (decided if it is true or false) in a way that every more precise structure will assign the same truth value. At the end of the chapter, we propose some extensions to $\mathcal{COL}_m$. We show how the modal axioms can be imposed on the logic and we define $\mathcal{COL}_m^+$: $\mathcal{COL}_m$ extended with public announcements. As in [Belle and Lakemeyer, 2015] we illustrate $\mathcal{COL}_m^+$ by formalising the muddy children puzzle.

6.1 A multi-agent Modal Logic accommodating Common Knowledge

We will here describe syntax and semantics of $\mathcal{CL}_m$, the first part of our logic. We will use this to give the main idea behind the proposed new semantical structures: μ-worlds.

As before, we will assume a fixed vocabulary Σ, a fixed set of agents $\mathcal{A}$ and a domain $D \subset \Sigma$, where each $d \in D$ is a constant symbol whose interpretation in all structures is d. This implies that every agent knows the domain to be D and the interpretation of every constant d to be d.

This also implies that the Barcan formula [Barcan, 1946]

$$K\forall x\varphi \Leftrightarrow \forall x K\varphi$$

is satisfied. However, it is well known how to relax this assumption by introducing predicates to express the domain as a subset of D [Fitting, 1999].

6.1.1 Syntax

Definition 6.1.1. We define the language $\mathcal{CL}_m$ by structural induction with the standard recursive rules of first order logic, augmented with:

$$K_A(\psi) \text{ is a formula if } \psi \text{ is a formula and } A \in \mathcal{A}$$

$$E_G(\psi) \text{ is a formula if } \psi \text{ is a formula and } G \subseteq \mathcal{A}$$

$$C_G(\psi) \text{ is a formula if } \psi \text{ is a formula and } G \subseteq \mathcal{A}$$

When $G = \mathcal{A}$, we use E and C as syntactic sugar for respectively $E_\mathcal{A}$ and $C_\mathcal{A}$.

Intuitively $K_A\psi$ can be read as "*A knows* ψ, $E\psi$ means that "*everybody knows* ψ and we can read $C\psi$ as "ψ is common knowledge". The difference between everybody knowing something and common knowledge is that common knowledge also captures information on what every agent knows about the knowledge of others. If p is common knowledge, then everybody knows p (Ep), everybody knows that everybody knows p, everybody knows that everybody knows that everybody knows p, ... As such, common knowledge is a much stronger notion than everybody knows.

We call formulas in $\mathcal{CL}_m$ that do not contain any occurrences of K_A, E_G or C_G *modal-free*, or *objective*, and we refer to an atom of the form $\chi\varphi$ ($\chi \in \{K_A, E_G, C_G\}$) as a *modal atom.* We say that φ is a modal component of ψ if ψ contains a modal atom $\chi\varphi$.

We illustrate the syntax by translating a few sentences to $\mathcal{CL}_m$.

Example 6.1.2. For these examples about the Watergate scandal (as in [Fagin et al., 1995]), we use a set of agents $\mathcal{A} = \{D, N\}$ and a vocabulary Σ:

$$\begin{aligned}
&\Sigma = \{\\
&\quad \Sigma_T = \{person\}\\
&\quad \Sigma_P = \{Burgles(person, person),\ President(person)\}\\
&\quad \Sigma_F = \{McCord : person,\ OBrien : person,\ Nixon : person\}\\
&\}
\end{aligned}$$

- Nixon does not believe that Dean knows that he is president

$$\neg K_N K_D President(Nixon)$$

- Everybody knows that Nixon is president, but it is not common knowledge.

$$E\ (President(Nixon)) \land \neg C\ (President(Nixon))$$

- Dean does not know whether Nixon knows that Dean knows that Nixon knows that McCord burgled O'Brien's office at Watergate.

$$\begin{aligned}
&\neg K_D(K_N K_D K_N Burgles(McCord, OBrien))\\
&\qquad \land \neg K_D(\neg K_N K_D K_N Burgles(McCord, OBrien))
\end{aligned}$$

In the previous chapter, the knowledge of each agent A was characterized by a specific theory. The agents modeled in dAEL were completely introspective and we assumed open and honest communication between the agents. In $\mathcal{CL}_m$, we look at the world from a "god perspective" i.e. from the view of an external viewer who describes the world in a theory and assume no modal axioms such as introspection and the truthfulness of the knowledge of an agent.

Example 6.1.3. In dAEL the third sentence

$$\begin{aligned}
&\neg K_D(K_N K_D K_N Burgles(McCord, OBrien))\\
&\qquad \land \neg K_D(\neg K_N K_D K_N Burgles(McCord, OBrien))
\end{aligned}$$

would reduce to

$$\neg(K_N Burgles(McCord, OBrien)) \land \neg(\neg K_N Burgles(McCord, OBrien))$$

and thus further to false, because of the mutual introspection ($K_A K_B \varphi$ simplifies to $K_B \varphi$ and $K_A \neg K_B \varphi$ simplifies to $\neg K_B \varphi$ for any formula φ and agents A, B).

6.1.2 Semantics

In standard first-order logic, a structure gives a formal account of the state of affairs. In this sense, a structure plays the same role in Tarskian model semantics for first order logic as worlds do in modal logic. In a context where the world contains a set of agents, what is missing in such FO structures is an account of the epistemic state of the agents in that world. Our aim here is to extend structures to provide such an account.

In order to do this, we introduce μ-worlds, defined for arbitrary ordinals μ. Intuitively, a μ-world is:

- A structure as defined in first-order logic if $\mu = 0$. When $\mu = 0$, a μ-world w accounts for a state of affairs of the objective world.
- A 0-world, with a set of $(\mu - 1)$-worlds for each agent A, if μ is a successor ordinal. This way it captures the state of affairs in the objective world and gives an account for every agent's belief (by stating the possible worlds) up to level $\mu - 1$ (i.e., $\mu - 1$ nested references to his own or an other agent's knowledge).
- A sequence of worlds $w = \langle v_\alpha \rangle_{\alpha < \mu}$ of length μ, where each v_α is an α-world if μ is a limit ordinal. This way, w contains information on the state of the world up to depth α, for any $\alpha < \mu$. Note that to make such a sequence make sense, we need some extra constraints. An α-world in the sequence should extend all earlier worlds, it should not, for example, have a different objective world. We formalize this idea in definition 6.1.8 using the concept of consistent worlds.

Definition 6.1.4. The set of μ-worlds $\mathcal{W}_\mu$ over a vocabulary Σ and a set of agents $\mathcal{A}$ is defined by induction [1].:

- A 0-world w is a Σ-structure S (with domain D) as defined in first-order logic.
- A $\mu + 1$-world w is a tuple consisting of a 0-world and for each agent $A \in \mathcal{A}$ a set of μ-worlds, which we will denote as A^w.
- A λ-world w (λ is a limit ordinal) is a sequence $\langle v_0, \ldots, v_\alpha, v_{\alpha+1}, \ldots \rangle$ of length λ, where each $v_\alpha \in \mathcal{W}_\alpha$, for $\alpha < \lambda$.

[1] We will from here on use k, l and κ to address successor ordinals, λ to address limit ordinals and μ as an arbitrary ordinal. We will not state this explicitly each time.

Given a $\mu + 1$-world w, w^{obj} is denoted as the objective state of w, and we call A^w the epistemic state of agent A in world w. The notation of the epistemic state A^w is based on the idea that for a symbol σ it is standard to denote the value of σ in w as σ^w, and the epistemic state A^w can be seen as the value of agent A in w. Since a λ-world is a sequence of worlds, we will use standard notation and given a λ-world w, we use $(w)_\alpha$ to denote element α of sequence w, which is an α-world. Given a λ-world w, w^{obj} is also denoted as the objective state of w and is defined as $w^{obj} = (w)_0$.

Example 6.1.5. In Example 6.1.2 we have shown some examples on the Watergate scandal. A 2-world representing a possible state of affairs for that vocabulary is shown in Figure 6.1. This figure will be used in a later example and the semantics will be illustrated there.

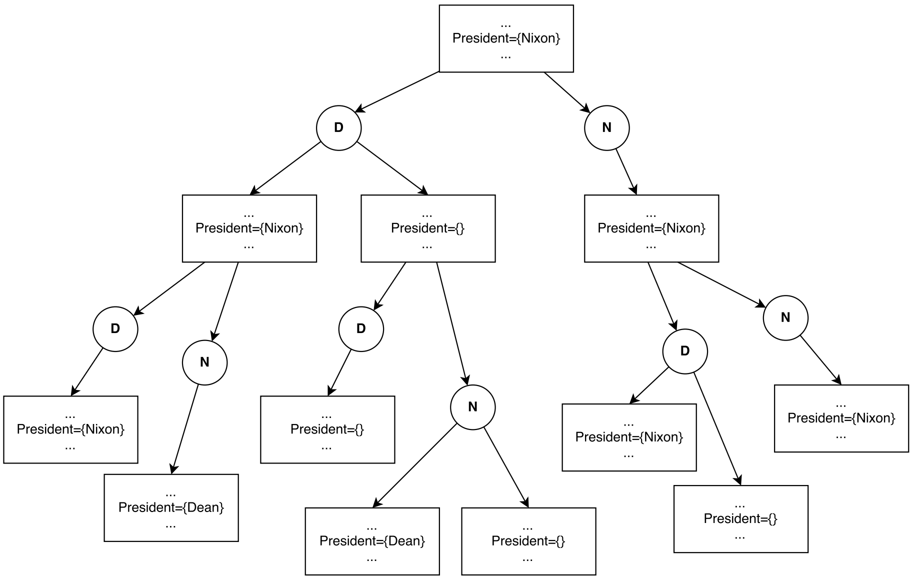

Figure 6.1: A possible 2-world for the Watergate-example.

Definition 6.1.6. We define, for each ordinal μ the **reduction** R_μ as a function $\mathcal{W}_{\mu+1} \to \mathcal{W}_\mu$ per induction on μ:

- For all $w \in \mathcal{W}_1$
$$R_0(w) = w^{obj}$$
- For all $w \in \mathcal{W}_{\mu+1}$, μ a successor ordinal:
$$R_\mu(w) = (w^{obj}, (R_{\mu-1}[A^w])_{A \in \mathcal{A}})$$

- For all $w \in \mathcal{W}_{\mu+1}$, μ a limit ordinal:

$$R_\mu(w) = \langle v_{\mu'} \rangle_{\mu' < \mu}$$

  where

  - $v_0 = w^{obj}$
  - $v_{\mu'+1} = (w^{obj}, ([A^w]_{\mu'})_{A \in \mathcal{A}})$ for $\mu' + 1 < \mu$ a successor ordinal
  - $v_\lambda = \langle v_i \rangle_{i<\lambda}$.

  Where we use shorthands $[W]_\alpha$ and $R[W]$ for $\{(w)_\alpha | w \in W\}$ respectively $\{R(w) | w \in W\}$, with W a set of worlds.

Finally, we define R as the union of all these functions. R is now a function that is defined for all $(\mu + 1)$-worlds.

Definition 6.1.7. Given a μ-world w and a μ'-world w' with $\mu' \leq \mu$, we inductively define w to be more precise than w' (denoted $w \geq_p w'$) as follows:

- If $\mu = \mu'$: $w \geq_p w'$ iff $w = w'$
- If μ is a successor ordinal and $\mu > \mu'$: $w \geq_p w'$ iff $R(w) \geq_p w'$
- If μ is a limit ordinal and $\mu > \mu'$: $w \geq_p w'$ iff $w' = (w)_{\mu'}$

Definition 6.1.8. Finally, we define the class of consistent μ-worlds:

- a 0-world is consistent;
- a $\mu+1$-world w is consistent if for every A, every world in A^w is consistent;
- a λ-world $w = \langle w_\alpha \rangle_{\alpha<\lambda}$ is consistent if for each $\mu < \lambda$: $(w)_\mu$ is consistent and for each $\mu < \mu' < \lambda$: $(w)_\mu \leq_p (w)_{\mu'}$.

Next, we define the α-extraction X_α $(\alpha \leq \mu)$ from a μ-world.

Definition 6.1.9. Given a μ-world w, and an $\alpha \leq \mu$, we define the α-extraction $X_\alpha(w)$ as the unique α-world that w reduces to, by induction on μ:

- If $\mu = \alpha$ (which must be the case if $\mu = 0$): $X_\alpha(w) = w$.
- If μ is a successor ordinal and $\mu > \alpha$: $X_\alpha(w) = X_\alpha(R(w))$
- If μ is a limit ordinal and $\mu > \alpha$: $X_\alpha(w) = (w)_\alpha$

To explain the concept of a consistent λ-world, consider that any μ-world is an approximation of a possible state of affairs. By increasing μ, the granularity increases; the precision of our approximations increases. In particular, for every μ-world w and $\alpha < \mu$, the α-world $X_\alpha(w)$ is a less precise approximation of the same state of affairs as w itself. A λ-world w is to be a sequence of increasing precision; therefore, for each ordinal $\mu < \lambda$, $(w)_\mu$ must have following property:

Proposition 6.1.10. *For each μ-world w, for each pair $0 \leq \alpha \leq \beta \leq \mu$, $X_\alpha(X_\beta(w)) = X_\alpha(w)$.*

Proof. We prove this by transfinite induction on μ:

- If $\mu = \beta$ (which must be the case if $\mu = 0$) then $X_\beta(w) = w$, so

$$X_\alpha(X_\beta(w)) = X_\alpha(w)$$

- If μ is a successor ordinal and $0 \leq \alpha \leq \beta < \mu$, then $X_\alpha(w) = X_\alpha(R(w))$, and by induction, for each $\alpha \leq \beta < \mu - 1$:

$$X_\alpha(X_\beta(R(w))) = X_\alpha(R(w)) = X_\alpha(w)$$

  By definition of X_β we know that $X_\beta(R(w)) = X_\beta(w)$, or

$$X_\alpha(X_\beta(w)) = X_\alpha(w) = X_\alpha(w)$$

  Which proves that

$$X_\alpha(X_\beta(w)) = X_\alpha(w)$$

- If μ is a limit ordinal and $0 \leq \alpha \leq \beta < \mu$, then $X_\alpha(w) = (w)_\alpha$ and $X_\beta(w) = (w)_\beta$. We prove that $X_\alpha((w)_\beta) = (w)_\alpha$ by induction on β:
  - If $\beta = 0$, then $\alpha = 0$ and $X_\alpha((w)_\beta) = (w)_\alpha = w^{obj}$
  - If β is a successor ordinal, then

$$X_\alpha((w)_\beta) = X_\alpha(R((w)_\beta)) = X_\alpha((w)_{\beta-1}) = (w)_\alpha$$

  - If β is a limit ordinal, then

$$X_\alpha((w)_\beta) = ((w)_\beta)_\alpha = (w)_\alpha$$

□

Proposition 6.1.11. *If w is a consistent μ-world, then so is $X_\alpha(w)$ for each $\alpha < \mu$.*

Proof. Take $X_\alpha(w)$, for $\alpha < \mu$. We prove that $X_\alpha(w)$ is consistent by induction on μ:

- If $\mu = 0$, the proposition is trivial.
- If μ is a successor ordinal,
$$X_\alpha(w) = X_\alpha(R(w))$$
and since $R(w)$ is $(\mu-1)$-world, we know that $X_\alpha(R(w))$ is consistent by the induction hypothesis.
- If μ is a limit ordinal,
$$X_\alpha(w) = (w)_\alpha$$
so $X_\alpha(w)$ is consistent by induction hypothesis.

□

Proposition 6.1.12. *For each $\alpha < \mu$, each α-world w has at least one refining μ-world $w' \geq_p w$.*

Proof. Take an α-world w and $\mu > \alpha$. We prove that there is a μ-world w' such that $X_\alpha(w') = w$ (then, by construction $w' \geq_p w$) by induction on μ:

- If $\mu = 0$, the proposition is trivial.
- If μ is a successor ordinal, then take w^* a $(\mu-1)$-world such that $X_\alpha(w^*) = w$ (the fact that this exists follows from the induction hypothesis) and define $w' = (w^{obj}, (A^{w'})_{A\in\mathcal{A}})$, where for each $A \in \mathcal{A}$: $A^{w'} = \{v | X_{\mu-1}(v) \in A^{w^*}\}$. Then $R(w') = w^*$, so $X_\alpha(w') = X_\alpha(w^*) = w$.
- If μ is a limit ordinal, define $w' =< v_{\mu'} >$ with
  - $v_{\mu'} = X_\alpha(w)$ if $\mu' < \alpha$
  - $v_{\mu'} = w$ if $\mu' = \alpha$
  - If $\alpha < \mu' < \mu$: $v_{\mu'} = w''$, where w'' is a world such that $X_\alpha(w'') = w$.

  It is clear that w' is consistent and the fact that $X_\alpha(w')$ now follows from the induction hypothesis.

□

From now on, we consider only consistent μ-worlds. That is, when in the sequel, we refer to μ-worlds, we mean consistent μ-worlds.

Proposition 6.1.13. *For each w an α-world, w' a μ-world with $w' \geq_p w$ and for each agent A: For every $v \in A^w$ there is a $v' \in A^{w'}$ such that $v' \geq_p v$.*

Proof. Take an α-world w and a μ-world w' with $w' \geq_p w$. Also take an agent A, and $v \in A^w$, we prove that there is a $v' \in A^{w'}$ such that $v' \geq_p v$ by induction on μ and α:

- If $\mu = 2$, then v is a 0-world. Since $w' \geq_p w$, we know that $R(w') = w$ and since $R[A^{w'}] = A^{R(w')} = A^w$, there is a $v' \in A^{w'}$ such that $v' \geq_p v$.
- Assume μ is a successor ordinal. Since $w' \geq_p w$, we know that $R(w') \geq_p w$ and we use the induction hypothesis there is a $v^* \in A^{R(w')}$ such that $v^* \geq_p v$. But, since $R[A^{w'}] = A^{R(w')}$, there is a $v' \in A^{w'}$ such that $v' \geq_p v^* \geq_p v$.
- Assume μ is a limit ordinal. Then construct v' as follows: For each $\alpha' < \alpha$ there is an α' world $v^*_{\alpha'}$ such that v
  - $(v')_{\alpha'} = v_{\alpha'}$ for all $\alpha' \leq \alpha$.
  - For each $\alpha' > \alpha$:
    - If α' is a successor ordinal take $(v')_{\alpha'}$ any world such that $R((v')_{\alpha'}) = (v')_{\alpha'-1}$
    - If α' is a limit ordinal take $(v')_{\alpha'} =< v'_{\alpha''} >_{\alpha''<\alpha'}$.

It is easy that v' is consistent and $v' \in A^{w'}$ follows from construction. □

Definition 6.1.14. Given a $\varphi \in \mathcal{CL}_m$ and a μ-world w, we define the three-valued valuation function φ^w by induction on μ:

If $\mu = 0$ or a successor ordinal, we define φ^w by induction on the structure of φ:

$$\begin{aligned}
P(\bar{t})^w &= \quad (\bar{t} \in P^{w^{obj}}) \\
(\varphi \wedge \psi)^w &= \quad glb_{\leq_t}(\varphi^w, \psi^w) \\
(\neg\varphi)^w &= \quad (\varphi^w)^{-1} \\
(\forall x : \varphi)^w &= \quad glb_{\leq_t}\{\varphi[x/d]^w \mid d \in D\} \\
(K_A\varphi)^w &= \quad \begin{cases} \mathbf{u} & \text{if } w \text{ is a 0-world} \\ glb_{\leq_t}\{\varphi^{w'} | w' \in A^w\} & \text{otherwise} \end{cases} \\
(E_G\varphi)^w &= \quad \begin{cases} \mathbf{u} & \text{if } w \text{ is a 0-world} \\ glb_{\leq_t}\{\varphi^{w'} | w' \in A^w \text{ and } A \in G\} & \text{otherwise} \end{cases} \\
(C_G\varphi)^w &= \quad glb_{\leq_t}\{(E_G^k\varphi)^w | k \in \mathbb{N}_0\}
\end{aligned}$$

where we use $E_G^k\varphi$ as a shorthand for k nestings of E_G:

$$E_G(E_G(E_G(\dots(\varphi))))$$

If μ is a limit ordinal, we define φ^w:

$$\varphi^w = lub_{\leq_p}\{\varphi^{(w)_{\mu'}} | \mu' < \mu\}$$

We say a μ-world w *satisfies a formula* φ (notation: $w \models \varphi$) if $\varphi^w = \mathbf{t}$. A μ-world w satisfies, or is a model of, a theory if it satisfies all formulas in that theory.

Proposition 6.1.15. *For each ordinal μ and μ-world w:*

$$w \models E_G\varphi \Leftrightarrow \bigwedge_{A \in G} K_A\varphi$$

Proof. We prove this by induction on μ.

- If $\mu = 0$, $(E_G\varphi)^w = \mathbf{u} = (\bigwedge_{A\in G} K_A)^w$ for any $w \in \mathcal{W}_0$.
- If $\mu = \alpha + 1$, take $w \in \mathcal{W}_{\alpha+1}$. The fact that $(E_G\varphi)^w = (\bigwedge_{A\in G} K_A)^w$ follows directly from the definition from the truth valuation for a successor ordinal.

- If $\mu = \lambda$ is a limit ordinal, take $w \in \mathcal{W}_\lambda$. Since $(\psi)^w = lub_{\leq_p}\{\psi^{(w)_\alpha} | \alpha < \lambda\}$ for any formula ψ and $(E_G\varphi)^{(w)_\alpha} = (\bigwedge_{A \in G} K_A)^{(w)_\alpha}$ for any $\alpha < \lambda$ by induction, it follows that $(E_G\varphi)^w = (\bigwedge_{A \in G} K_A)^w$.

$\square$

Example 6.1.16. Continuing Example 6.1.2, we look at the second sentence discussed there: "Nixon is not sure that Dean knows that he is president". In $\mathcal{CL}_m$, with the vocabulary as defined in Example 6.1.2 this sentence can be written as:

$$\neg K_N K_D President(Nixon)$$

We look at two 2-worlds, in figure 6.2 and 6.3. For readability, we will only show the relevant part for the structures in the 2-worlds.

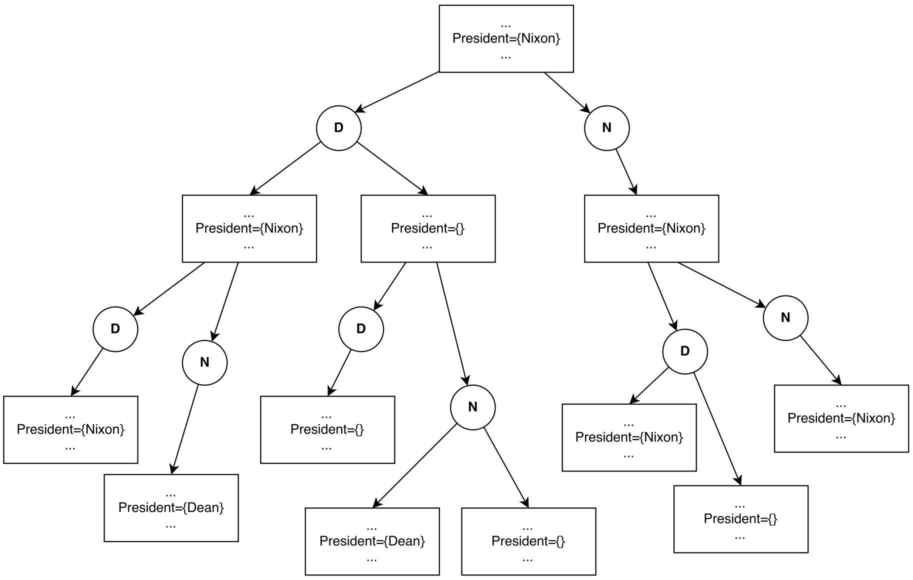

Figure 6.2: World 1 for Example 6.1.16

The root of the tree in the figures is a structure w^{obj} representing the objective state of the world. Agents are represented by a circular node (D for *Dean*, N for *Nixon*), and have edges to all 1-worlds that are possible according to them. These 1-worlds contain on their turn an objective world and a epistemic state (a set of 0-worlds) for each agent.

In world 1, the sentence $\neg K_N K_D President(Nixon)$ is true, while it is not in world 2. Next to this, $E(President(Nixon))^{w_2} = \mathbf{t}$ and $E^2(President(Nixon))^{w_2} =$

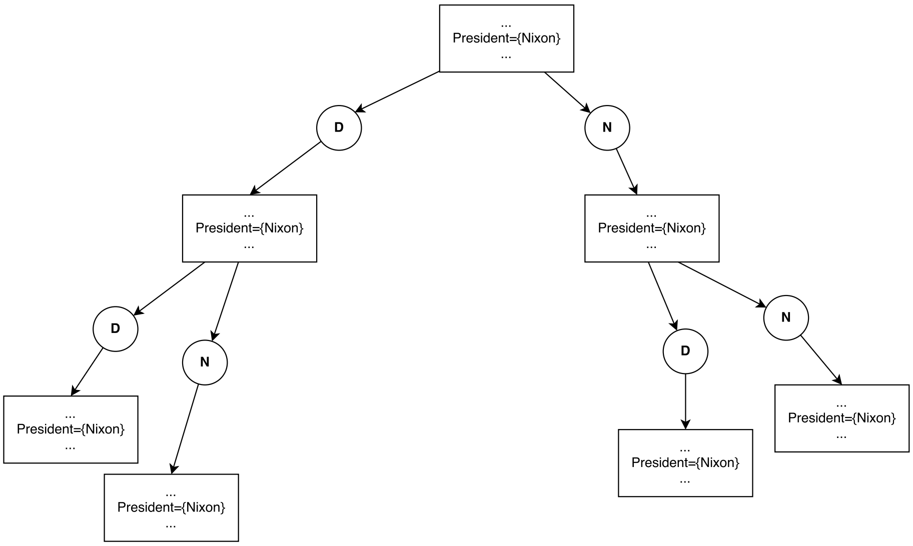

Figure 6.3: World 2 for Example 6.1.16

t in world 2, but note that $C(President(Nixon))$ is not true in world 2, this is unknown because this world is not deep enough to capture common knowledge.

6.1.3 Resolution of a logic

We study the behaviour of these new logics and the worlds, by looking at the connections between the depth of the world, reductions, the set of formulas that the world can resolve (the set of formulas φ for w, where $\varphi^w \neq \mathbf{u}$) and the set of satisfiable formulas in that logic.

Definition 6.1.17 (Resolves)**.** A μ-world w *resolves* a formula φ, if $\varphi^w \neq \mathbf{u}$, it resolves a language $\mathcal{L}$ if it resolves every sentence $\varphi \in \mathcal{L}$.

Proposition 6.1.18. *For any μ-world w and any formula φ in $\mathcal{CL}_m$:*

$$\varphi^{R(w)} \leq_p \varphi^w$$

Proof. We prove that for any arbitrary $\mu'+1$-world w and any formula $\varphi \in \mathcal{CL}_m$: $\varphi^w \geq_p \varphi^{R(w)}$. We distinguish three cases:

1. If $\varphi^{R(w)} = \mathbf{u}$, it is trivial that $\varphi^w \geq_p \varphi^{R(w)}$, since every truth value is more (or as) precise as $\mathbf{u}$.

2. If $\varphi^{R(w)} = \mathbf{t}$, we prove that $\varphi^w = \mathbf{t}$.
3. If $\varphi^{R(w)} = \mathbf{f}$, we prove that $\varphi^w = \mathbf{f}$. The proof for this is completely symmetrical to case 2.

We will only prove case 2, so assume an arbitrary $\varphi \in \mathcal{CL}_m$ such that $\varphi^{R(w)} = \mathbf{t}$. We prove that $\varphi^w = \mathbf{t}$ by induction on the structure of φ.

- If $\varphi = P(\bar{t})$ then $\varphi^{R(w)} = \varphi^{R(w)^{obj}} = \varphi^{w^{obj}} = \varphi^w$.
- If $\varphi = \varphi_1 \wedge \varphi_2$, then $\varphi^w = glb_{\leq_t}(\varphi_1^w, \varphi_2^w)$ and by induction $\varphi_i^w \geq_p \varphi_i^{R(w)}$. It follows that $\varphi^w = glb_{\leq_t}(\varphi_1^w, \varphi_2^w) \geq_p glb_{\leq_t}(\varphi_1^{R(w)}, \varphi_2^{R(w)}) = \varphi^{R(w)}$.
- If $\varphi = \varphi_1 \vee \varphi_2$, $\varphi = \neg\psi$, $\varphi = \forall x\psi$, ..., the proof is the similar as above.
- If $\varphi = K_A\psi$ then we prove that $\psi^{w'} = \mathbf{t}$ for every $w' \in A^w$. As such, we fix an arbitrary $w' \in A^w$ and prove that $w' \vDash \psi$. Since $R(w') \in R[A^w]$ and $R[A^w] = A^{R(w)}$, it follows that $\psi^{R(w')} = \mathbf{t}$. And by induction hypothesis ($\psi^{R(w')} \leq_p \psi^{w'}$) we can derive that $\psi^{w'} = \mathbf{t}$.
- If $\varphi = E_G\psi$, $\varphi^w = \mathbf{t}$ follows directly from the fact that $E_G\psi$ is syntactic sugar for $\bigwedge_{A\in G} K_A\psi$ and the proof above.
- If $\varphi = C_G\psi$, we prove that for each $k \in \mathbb{N}_0$:

  $$(E_G^k\psi)^w = \mathbf{t}$$

  We prove this by induction on $k \in \mathbb{N}_0$:

  – For $k = 1$, it follows from the the previous case that $\mathbf{t} = (E_G\psi)^{R(w)} = \varphi^{R(w)} \leq_p = \varphi^w = (E_G\psi)^w = \mathbf{t}$.

  – For $k = n + 1$, we find that $\varphi = (E_G^{n+1}\psi) = (E_G(E_G^n\psi))$ and use the induction hypothesis together with the previous case to find that $\mathbf{t} = (E_G(E_G^n\psi))^{R(w)} \leq_p (E_G(E_G^n\psi))^w = \varphi^w$.

This proves that $\varphi^w = \mathbf{t}$ for all $\varphi \in \mathcal{CL}_m$. □

A logic is defined as the combination of a language (syntax) with a semantics (a truth valuation). We define for every ordinal μ the logic μ-$\mathcal{CL}_m$ as the logic with language $\mathcal{CL}_m$ and the valuation as defined above, using μ-worlds. Before we do this, we define a few intermediate concepts such as the set of satisfiable formulas within that logic:

$$Sat(\mu\text{-}\mathcal{CL}_m) = \{\varphi \in \mathcal{CL}_m | \exists w \in \mathcal{W}_\mu : \ w \models \varphi\}$$

We now define the concept of a resolution of a logic: the set of pairs of distinguishable formulas within that logic.

Definition 6.1.19. A pair of formulas φ_1 and φ_2 is in the resolution of a logic μ-$\mathcal{CL}_m$ (notation: $(\varphi_1, \varphi_2) \in Res(\mu\text{-}\mathcal{CL}_m)$) if there is a μ-world w such that $w \models \varphi_1$ and $w \not\models \varphi_2$.

The name resolution of a logic is based on the resolution of a microscope. A microscope A has a higher resolution than a microscope B if there is a pair of points between which A can distinguish, while B cannot. For a logic, we say that a logic A has a (not necessarily strict) higher resolution than a logic B if A can distinguish between all formulas between which B can distinguish. This is closely related with the set of satisfiable formulas of a logic.

Proposition 6.1.20. *For any two logics μ-$\mathcal{CL}_m$ and μ'-$\mathcal{CL}_m$ and for any two formulas $\varphi_1, \varphi_2 \in \mathcal{CL}_m$ it is true that*

$$Res(\mu'\text{-}\mathcal{CL}_m) \subseteq Res(\mu\text{-}\mathcal{CL}_m) \textit{ iff } Sat(\mu'\text{-}\mathcal{CL}_m) \subseteq Sat(\mu\text{-}\mathcal{CL}_m)$$

Proof. Take two logics μ-$\mathcal{CL}_m$ and μ'-$\mathcal{CL}_m$.

Assume $Sat(\mu'\text{-}\mathcal{CL}_m) \subseteq Sat(\mu\text{-}\mathcal{CL}_m)$. Now take $(\varphi_1, \varphi_2) \in Res(\mu'\text{-}\mathcal{CL}_m)$. Then, by definition $\varphi_1 \wedge \neg\varphi_2 \in Sat(\mu'\text{-}\mathcal{CL}_m)$, and thus $\varphi_1 \wedge \neg\varphi_2 \in Sat(\mu\text{-}\mathcal{CL}_m)$, or $(\varphi_1, \varphi_2) \in Res(\mu\text{-}\mathcal{CL}_m)$.

Assume $Res(\mu'\text{-}\mathcal{CL}_m) \subseteq Res(\mu\text{-}\mathcal{CL}_m)$. Now take $\varphi_1 \in Sat(\mu'\text{-}\mathcal{CL}_m)$, then $(\varphi_1, p \wedge \neg p) \in Res(\mu'\text{-}\mathcal{CL}_m)$ for any propositional variable p. This, $(\varphi_1, p \wedge \neg p) \in Res(\mu\text{-}\mathcal{CL}_m)$, or $\varphi_1 \in Sat(\mu\text{-}\mathcal{CL}_m)$. ☐

Proposition 6.1.21. *For any μ-world w and μ'-world w', where $w' \geq_p w$ and any formula $\varphi \in \mathcal{CL}_m$:*

$$\varphi^{w'} \geq_p \varphi^w$$

Proof. Take an arbitrary μ-world w, μ'-world w', $w' \geq_p w$ and $\mathcal{CL}_m$ formula φ. We know that $\mu' \geq \mu$ since $w' \geq_p w$. We prove that $\varphi^{w'} \geq_p \varphi^w$ by transfinite induction on μ'.

- If $\mu' = \mu$, $w' \geq_p w$ only if $w' = w$.
- If μ' is a successor ordinal $> \mu$, we know that $\varphi^{R(w')} \geq_p \varphi^w$ by induction. Using Proposition 6.1.18, it follows that $\varphi^{w'} \geq_p \varphi^{R(w')} \geq_p \varphi^w$.

- If μ' is a limit ordinal $> \mu$, the monotonicity follows directly from the definition of the truth valuation for μ' a limit ordinal:

$$\varphi^{w'} = lub_{\leq_t}\{\varphi^{(w')_\alpha} | \alpha < \mu'\}$$

  and the fact that we know that $\varphi^{(w')_\alpha} \geq_p \varphi^w$ for each $\alpha < \mu'$ by induction.

This proves that $\varphi^{w'} \geq_p \varphi^w$ for all w, w' where $w' \geq_p w$ and all $\varphi \in \mathcal{CL}_m$. $\square$

We now define the modal depth of formula, that states how deep a formula refers to other agent's knowledge

Definition 6.1.22. The *modal depth* $MD(\varphi)$ of a formula $\varphi \in \mathcal{CL}_m$ is defined by induction on the structure of φ:

- $MD(P(\bar{t})) = 0$
- $MD(\neg\varphi) = MD(\varphi)$
- $MD(\varphi \wedge \psi) = max(MD(\varphi), MD(\psi))$
- $MD(\forall x : \varphi) = MD(\varphi)$
- $MD(K_A\varphi) = MD(\varphi) + 1$
- $MD(E_G\varphi) = MD(\varphi) + 1$
- $MD(C_G\varphi) = \nu$ with ν being the smallest limit ordinal $\geq MD(\varphi)$.

Proposition 6.1.23. *Given a $\mathcal{CL}_m$ formula φ_1 with subformula φ_2, then $MD(\varphi_2) \leq MD(\varphi_1)$.*

Proof. This is easy to prove by induction on the structure of φ_1. $\square$

Proposition 6.1.24. *For every formula φ in $\mathcal{CL}_m$: $MD(\varphi) < \omega^2$.*

Proof. This follows directly from the fact that an operator can only highten the modal depth of a formula by ω and a formula is of finite length. $\square$

Theorem 6.1.25. *A μ-world w (μ an arbitrary ordinal) can resolve every formula $\varphi \in \mathcal{CL}_m$, with $MD(\varphi) < \mu$.*

Proof. We prove that a μ-world resolves all $\mathcal{CL}_m$ formulas with modal depth $< \mu$. We prove this by induction on μ.

First, assume that μ is zero, or is a successor ordinal. Take a μ-world w, and a formula $\varphi \in \mathcal{CL}_m$, with $MD(\varphi) < \mu$. We prove that $\varphi^w \neq \mathbf{u}$ by induction on the structure of φ. So, assume that $\varphi'^w \neq \mathbf{u}$ for all φ' that are subformulas of φ.

- If $\varphi = P(\bar{t})$ then $\varphi^w = \varphi^{w^{obj}} \neq \mathbf{u}$.
- If $\varphi = \varphi_1 \wedge \varphi_2$, note that $\varphi^w = glb_{\leq_t}(\varphi_1^w, \varphi_2^w)$ and since by induction neither $\varphi_1^w = \mathbf{u}$ nor $\varphi_2^w = \mathbf{u}$, it follows that $\varphi^w \neq \mathbf{u}$.
- If $\varphi = \varphi_1 \vee \varphi_2$, $\varphi = \neg\psi$, $\varphi = \forall x \psi$, ..., the proof is similar.
- If $\varphi = K_A\psi$, we know that w cannot be a 0-world, since w is a μ-world, with $\mu \geq MD(\varphi) = MD(\psi) + 1 > 0$. Since by induction it follows that for each $w' \in A^w$ ($(\mu - 1)$-worlds): $\psi^{w'} \neq \mathbf{u}$, we know that $(K_A\psi)^w = glb_{\leq_t}\{\psi^{w'} | w' \in A^w\} \neq \mathbf{u}$.
- If $\varphi = E_G\psi$, $\varphi^w \neq \mathbf{u}$ follows directly from the fact that $E_G\psi$ is syntactic sugar for a conjunction $\bigwedge_{A \in G} K_A\psi$ and the proof above.
- If $\varphi = C_G\psi$, we prove that $(C_G\psi)^w \neq \mathbf{u}$. Since $(C_G\psi)^w = glb_{\leq_t}\{(E_G^k\psi)^w | k \in \mathbb{N}_0\}$ it is sufficient to prove that $(E_G^k\psi)^w \neq \mathbf{u}$ for all $k \in \mathbb{N}_0$. This follows from the induction hypothesis since $C\psi^{R(w)} \neq \mathbf{u}$ implies that $(E_G^k\psi)^{R(w)} \neq \mathbf{u}$ for all $k \in \mathbb{N}_0$, which implies that $(E_G^k\psi)^w \neq \mathbf{u}$ (by induction hypothesis) for all k.

Now, assume μ is a limit ordinal. Take a μ-world w, and a formula $\varphi \in \mathcal{CL}_m$, with $MD(\varphi) < \mu$. We know that $MD(\varphi) < \mu$ and μ is a limit ordinal. Define $\alpha = MD(\varphi) + 1$, then $\alpha < \mu$. Using the induction hypothesis, we find that $\varphi^{(w)_\alpha} \neq \mathbf{u}$, and from the monotonicity of the valuation (Proposition 6.1.21) it follows that

$$\varphi^w = lub_{\leq_t}\{\varphi^{(w)_\alpha} | \alpha < \mu\} \neq \mathbf{u}$$

This proves that $\varphi^w \neq \mathbf{u}$ for all $\varphi \in \mathcal{CL}_m$. And by Proposition 6.1.21, it follows that $\varphi^{w'} \geq_p \varphi^w$. □

Consequence 6.1.1. *Given a μ-world w and a formula φ with $MD(\varphi) < \mu$, then for every μ'-world $w' \geq_p w$, we have $\varphi^w = \varphi^{w'}$.*

Proof. This follows directly from Proposition 6.1.21 and Theorem 6.1.25. □

Consequence 6.1.2. *Any ω^2-world resolves $\mathcal{CL}_m$.*

Proof. Follows directly from Proposition 6.1.24 and Theorem 6.1.25. □

This allows us to prove that the set of satisfiable formulas in μ-$\mathcal{CL}_m$ grows when μ becomes higher.

Theorem 6.1.26. *μ-$\mathcal{CL}_m$ has a (not necessarily strict) larger set of satisfiable formulas than μ'-$\mathcal{CL}_m$ if $\mu > \mu'$.*

Proof. Take μ and μ' arbitrary ordinals, such that $\mu > \mu'$ and $\varphi \in Sat(\mu'\text{-}\mathcal{CL}_m)$. We prove that $\varphi \in Sat(\mu\text{-}\mathcal{CL}_m)$.

If $\varphi \in Sat(\mu'\text{-}\mathcal{CL}_m)$, then there is a $w' \in \mathcal{W}_{\mu'}$ such that $w' \models \varphi$. Take $w \in \mathcal{W}_\mu$ such that $w \geq_p w'$, which exists (Proposition 6.1.12). Using Theorem 6.1.25, we find that $\varphi^w \geq_p \varphi^{w'}$, and by consequence that $\varphi \in Sat(\mu\text{-}\mathcal{CL}_m)$. □

Consequence 6.1.3. *There is no μ-$\mathcal{CL}_m$ logic such that $Sat(\omega^2\text{-}\mathcal{L}_m) \subsetneq Sat(\mu\text{-}\mathcal{CL}_m)$.*

Proof. Take a logic μ-$\mathcal{CL}_m$ such that

$$Sat(\omega^2\text{-}\mathcal{CL}_m) \subseteq Sat(\mu\text{-}\mathcal{CL}_m)$$

We prove

$$Sat(\omega^2\text{-}\mathcal{CL}_m) = Sat(\mu\text{-}\mathcal{CL}_m)$$

So take $\varphi \in Sat(\mu\text{-}\mathcal{CL}_m)$ and w such that $w \models \varphi$. By Proposition 6.1.24, $MD(\varphi) < \omega^2$. This means that $\varphi^{w'} \neq \mathbf{u}$, for any $w' \in \mathcal{W}_{\omega^2}$ (Theorem 6.1.25). So assume towards contradiction that $\varphi^{w'} = \mathbf{f}$, for any $w' \in \mathcal{W}_{\omega^2}$. But since for $w'' = X_{\omega^2}(w)$, it is the case that $\varphi^{w''} = \varphi^w$ (Theorem 6.1.25), it follows that $\varphi^w = \mathbf{f}$, which is a contradiction. So it must be the case that there is a ω^2-world such that $w'' \models \varphi$. □

The bound proposed in Theorem 6.1.25 (ω^2) is strict as we will show in the following proposition.

Proposition 6.1.27. *For every $k \geq 0$, there is a formula φ such that there is a world w in $\mathcal{W}_{k\omega}$: $\varphi^w = \mathbf{u}$*

Proof. Assume that $\Sigma = \{p\}$ and $\mathcal{A} = \{A\}$. Since $(K_A p) = \mathbf{u}$ for any $w \in \mathcal{W}_0$ by definition, we fix an arbitrary $k \in \mathbb{N}_0$. Now define a formula $\varphi = C^k p$, we prove that there is a $w \in \mathcal{W}_{k\omega}$ such that $(C^k p)^w = \mathbf{u}$ and $(C^{k-1}p)^w = \mathbf{t}$, by induction on k (and refer to this induction hypothesis as induction hypothesis 1).

Define a world $w_k \in \mathcal{W}_{k\omega}$ for each k as follows:

- $(w_k)_0 = \{p\}$
- $(w_k)_{\alpha+1} = \{p, (A^{(w_k)_{\alpha+1}})_{A\in\mathcal{A}}\}$, where for each $A \in \mathcal{A} : A^{(w_k)_{\alpha+1}} = \{(w_k)_\alpha\}$
- $(w_k)_\lambda = \langle v_\beta \rangle_{\beta<\alpha}$ where each $v_\beta = (w_k)_\beta$.

Then:

- If $k = 1$ and w_k is a ω-world, then $(C^0p)^{w_k} = p^{w_k} = \mathbf{t}$ and

$$\begin{aligned}(Cp)^{w_k} &= lub_{\leq_p}\{(Cp)^{(w_k)_\alpha}|\alpha < \omega\}\\ &= lub_{\leq_p}\{glb_{\leq_t}\{(E^np)^{(w_k)_\alpha}|n \in \mathbb{N}_0\}|\alpha < \omega\}\\ &= lub_{\leq_p}\{\mathbf{u}|\alpha < \omega\}\\ &= \mathbf{u}\end{aligned}$$

  where the second step follows from the fact that $(E^np)^{(w_k)_\alpha} = \mathbf{t}$ for all $0 < n \leq \alpha$ and $(E^np)^{(w_k)_\alpha} = \mathbf{u}$ for all $n > \alpha$.

- If $k = k' + 1$ ($k' > 0$) and w_k is a $(k\omega)$-world, then

$$\begin{aligned}(C^kp)^{w_k} &= lub_{\leq_p}\{(C^kp)^{(w_k)_\alpha}|\alpha < k\omega\}\\ &= lub_{\leq_p}\{glb_{\leq_t}\{(E^nC^{k'}p)^{(w_k)_\alpha}|n \in \mathbb{N}_0\}|\alpha < k\omega\}\end{aligned}$$

  We need to prove that $(C^kp)^{(w_k)_\alpha} = \mathbf{u}$ for all $\alpha < k\omega$. If we can prove it for any α' such that $k'\omega < \alpha' < (k'+1)\omega$, which are all successor ordinals, it follows that $(C^kp)^{(w_k)_\alpha} = \mathbf{u}$ for all $\alpha < k\omega$ by Proposition 6.1.21. So take $\alpha' = k'\omega + l$ with $l \in \mathbb{N}_0$ and define $w'_{(k',l)} = (w_k)_{k'\omega+l}$. We prove that $(C^kp)^{w'_{(k',l)}} = \mathbf{u}$ by proving that $(E^n(C^{k'}p))^{w'_{(k',l)}} = \mathbf{t}$ for all $0 < n \leq l$ and $(E^n(C^{k'}p))^{w'_{(k',l)}} = \mathbf{u}$ for all $n > l$. We prove this by induction on n (and refer to this induction hypothesis as induction hypothesis 2).

- If $n = 1$, then

$$\begin{aligned} E((C^{k'}p))^{w'_{(k',l)}} &= glb_{\leq_t}\{(C^{k'}p)^v | v \in A^{w'_{(k',l)}}\} \\ &= (C^{k'}p)^{w'_{(k',l-1)}} \\ &= \mathbf{t} \end{aligned}$$

  by induction hypothesis 1, since $v = w'_{(k',l-1)}$ by construction of w and $w'_{(k',l-1)} \geq_p w'_{(k',0)} = w_{k'}$.
- If $n > 1$ and $n \leq l$, then

$$\begin{aligned} E^n((C^{k'}p))^{w'_{(k',l)}} &= glb_{\leq_t}\{E^{n-1}((C^{k'}p))^v | v \in A^{w'_{(k',l)}}\} \\ &= E^{n-1}((C^{k'}p))^{w'_{(k',l-1)}} \\ &= \mathbf{t} \end{aligned}$$

  by induction hypothesis 2, since $n - 1 \leq l - 1$.
- If $n = 2$ and $l < n$ (so $l = 1$), then

$$\begin{aligned} E^2((C^{k'}p))^{w'_{(k',1)}} &= glb_{\leq_t}\{E((C^{k'}p))^v | v \in A^{w'_{(k',1)}}\} \\ &= E((C^{k'}p))^{w'_{(k',0)}} \\ &= glb_{\leq_t}\{((C^{k'}p))^{(w'_{(k',0)})_\alpha} | \alpha < k'\omega\} \\ &= \mathbf{u} \end{aligned}$$

  by induction hypothesis 1.
- If $n > 1$ and $l < n$, then

$$\begin{aligned} E^n((C^{k'}p))^{w'_{(k',l)}} &= glb_{\leq_t}\{E^{n-1}((C^{k'}p))^v | v \in A^{w'_{(k',l)}}\} \\ &= E^{n-1}((C^{k'}p))^{w'_{(k',l-1)}} \\ &= \mathbf{u} \end{aligned}$$

  by induction hypothesis 2, since $n - 1 < l - 1$.

The proof that $(C^{k'}p)^{(w_k)} = \mathbf{t}$, is similar.

This proves that $\varphi^{w_k} = \mathbf{u}$. □

In the next section we will introduce the only-knowing modality and we show how important it is that the depth of worlds is well-chosen.

6.2 A multi-agent modal logic combining Only-Knowing and common knowledge

Only knowing was first introduced as a modality in a logic by Levesque in [Levesque, 1990] as a way to capture the notion that the beliefs of an agent are precisely those that follow from his knowledge base. He introduced an O operator in a modal logic, in addition to the *knowing* operator K. Intuitively the O operator also fixes what an agent does not know. For example, if an agent's knowledge base is $K(p \vee q)$, this does not exclude that the agent knows p, while if an agent's knowledge base is $O(p \vee q)$, this is excluded, since $p \vee q$ is all that is known.

In this section, we will extend $\mathcal{CL}_m$ with an operator for only knowing O_A for each agent A. We will discuss possible semantics, discuss the relevant issues and study how to solve them.

Definition 6.2.1. Given a vocabulary Σ and an indexed set of agents $\mathcal{A}$, we define the language $\mathcal{COL}_m$ by structural induction with the rules of Definition 6.1.1, augmented with:

$$O_A(\psi) \in \mathcal{COL}_m \qquad\qquad \text{if } \psi \in \mathcal{COL}_m \text{ and } A \in \mathcal{A}$$

Before we define the valuation function, using λ-worlds, we show two other approaches. One using Kripke structures, based on our λ-worlds and a recent approach by Belle and Lakemeyer. We use these to show the importance of the depth of the worlds and the resolution of the logic.

6.2.1 Intermezzo: The Valuation function

Kripke structures for $\mathcal{COL}_m$

The valuation proposed in Definition 6.1.14 for $\mathcal{CL}_m$ is unconventional in the sense that it is three-valued, which is not standard for modal logics. Using this valuation, we proved two results at the end of last subsection. We proved that every ω^2-world w resolves every $\mathcal{CL}_m$ formula φ (i.e., $\varphi^w \neq \mathbf{u}$) and that there is no ordinal $\mu > \omega^2$ such that there is a formula φ that is satisfiable in a μ-world that is not satisfiable in a ω^2-world. In fact, it is the case that for every ordinal μ where every μ-world resolves $\mathcal{CL}_m$ it immediately follows that the set of satisfiable formulas for μ-$\mathcal{CL}_m$ is maximal, i.e., there is no μ' such that

μ'-$\mathcal{CL}_m$ has a strict superset of satisfiable formulas. Before we define a valuation for only-knowing in this framework, we will show that while this correspondence holds for the valuation defined in Definition 6.1.14 this correspondence is not trivial.

To this extent, we propose an alternative valuation using λ-worlds that is two-valued. We do this by defining an *epistemic state* of an agent A in a λ-world.

Definition 6.2.2. An epistemic state A^w of an agent A in a λ-world w is the set of all λ-worlds w' satisfying following conditions:

- For all ordinals $\mu < \lambda : (w')_\mu \in A^{(w)_{\mu+1}}$
- w' is still a *consistent* world:
  - $\mu' + 1 < \lambda$: $R((w')_{\mu'+1}) = (w')_{\mu'}$
  - For each limit ordinal $\lambda' < \lambda$:

    $$((w')_{\lambda'})_\mu = (w')_\mu$$

    for each $\mu < \lambda'$

This allows us to define a logic λ-$\mathcal{COL}_m^2$ with a 2-valued valuation, defined by:

Definition 6.2.3. The truth valuation $(\cdot)^{2v:w}$ of a $\mathcal{COL}_m$ sentence in logic λ-$\mathcal{COL}_m^2$, given a λ-world w is defined by induction on the structure of φ:

$$
\begin{aligned}
P(\bar{t})^{2v:w} = \quad & (\bar{t} \in P^{w^{obj}}) \\
(\varphi \wedge \psi)^{2v:w} = \quad & glb_{\leq_t}(\varphi^{2v:w}, \psi^{2v:w}) \\
(\neg\varphi)^{2v:w} = \quad & (\varphi^{2v:w})^{-1} \\
(\forall x : \varphi)^{w} = \quad & glb_{\leq_t}\{\varphi[x/d]^{2v:w} \mid x \in D\} \\
(K_A\varphi)^{2v:w} = \quad & glb_{\leq_t}\{\varphi^{2v:w'} | w' \in A^w\} \\
(E_G\varphi)^{2v:w} = \quad & glb_{\leq_t}\{\varphi^{2v:w'} | w' \in A^w \text{ and } A \in G\} \\
& = (\bigwedge_{A \in G} K_A\varphi)^{2v:w} \\
(C_G\varphi)^{2v:w} = \quad & glb_{\leq_t}\{(E_G^k\varphi)^{2v:w} | k \in \mathbb{N}_0\} \\
(O_A\varphi)^{2v:w} = \quad & \begin{cases} \mathbf{t} & \text{if } A^w = \{w' \in S | \varphi^{2v:w'} = \mathbf{t}\} \\ \mathbf{f} & \text{if } A^w \neq \{w' \in S | \varphi^{2v:w'} = \mathbf{t}\} \end{cases}
\end{aligned}
$$

and we use E and C as a shorthand for respectively $E_{\mathcal{A}}$ and $C_{\mathcal{A}}$. We write $w \models^{2v} \varphi$ if $\varphi^{2v:w} = \mathbf{t}$

This allows us to define a Kripke structure for every limit ordinal λ: the λ-canonical Kripke structure. The connection between worlds and Kripke structures was also investigated in [Fagin et al., 1995]. Based on their exposition, we will formalise this connection for every λ.

Definition 6.2.4. Given a logic $\lambda - \mathcal{COL}_m$, with λ a limit ordinal, we define the λ-canonical Kripke structure as: $U_\lambda = \{S, (\mathcal{K}_A)_{A\in\mathcal{A}}\}$ as follows:

- $S = \mathcal{W}_\lambda$.
- $\mathcal{K}_A$: A pair (w, w') is in $\mathcal{K}_A$ if $w' \in A^w$.

The proposed truth valuation is based on the the standard 2-valued satisfaction relation in Kripke structures [Hughes and Cresswell, 1996].

Definition 6.2.5. The truth valuation of a $\mathcal{COL}_m$ sentence φ: $U, w \vDash' \varphi$ in a Kripke structure U and a state w is defined by induction on the structure of φ:

$$\begin{aligned}
P(\bar{t})^{U,w} &= && (\bar{t} \in P^{w^{obj}}) \\
(\varphi \wedge \psi)^{U,w} &= && glb_{\leq_t}(\varphi^{U,w}, \psi^{U,w}) \\
(\neg\varphi)^{U,w} &= && (\varphi^{U,w})^{-1} \\
(\forall x : \varphi)^{w} &= && glb_{\leq_t}\{\varphi[x/d]^{U,w} \mid x \in D\} \\
(K_A\varphi)^{U,w} &= && glb_{\leq_t}\{\varphi^{U,w'} | w' \in \mathcal{K}_A\} \\
(E_G\varphi)^{U,w} &= && glb_{\leq_t}\{\varphi^{U,w'} | w' \in \mathcal{K}_A \text{ and } A \in G\} \\
& && = (\bigwedge_{A \in G} K_A\varphi)^{U,w} \\
(C_G\varphi)^{U,w} &= && glb_{\leq_t}\{(E_G^k\varphi)^{U,w} | k \in \mathbb{N}_0\} \\
(O_A\varphi)^{U,w} &= && \begin{cases} \mathbf{t} & \text{if } \mathcal{K}_A = \{w' \in S | \varphi^{U,w'} = \mathbf{t}\} \\ \mathbf{f} & \text{if } \mathcal{K}_A \neq \{w' \in S | \varphi^{U,w'} = \mathbf{t}\} \end{cases}
\end{aligned}$$

and we use E and C as a shorthand for respectively $E_{\mathcal{A}}$ and $C_{\mathcal{A}}$. We say that a Kripke structure U and a state w satisfies a sentence φ (denoted $U, w \models' \varphi$) if $\varphi^{U,w} = \mathbf{t}$

Valuation $(\cdot)^{2v:w}$ and $(\cdot)^{U,w}$ are equivalent as next proposition shows.

Proposition 6.2.6. *Given a $\mathcal{COL}_m$ formula φ and U the λ-canonical Kripke structure, we have that*

$$\varphi^{U,w} = \varphi^{2v:w}$$

for every λ-world w.

Proof. This follows directly from the definitions of the valuation functions $(\cdot)^{2v:w}$ and $(\cdot)^{U,w}$. □

Given valuation $\models'$, the next proposition states that for every λ, every world in the λ-canonical Kripke structure resolves $\mathcal{COL}_m$, i.e.,the truth value of every $\mathcal{COL}_m$ sentence is 2-valued. However, this does not mean that the set of $\mathcal{COL}_m$ sentences true in an element of this Kripke-structure is maximal, as we will see in Proposition 6.2.9.

Proposition 6.2.7. *The truth value of every $\mathcal{COL}_m$ sentence is 2-valued in U, w, with U the ω-canonical Kripke structure and $w \in \mathcal{W}_\omega$.*

Proof. Trivial. □

Before we show a counterexample that proves that the set of $\mathcal{CL}_m$-formulas that can be satisfied using the ω-canonical Kripke structure is not maximal (there is a λ such that the λ-canonical Kripke structure satisfies strictly more formulas than the ω-canonical Kripke structure), we show a few intermediate results that we will need for the proof.

Theorem 6.2.8. *For any λ, the 3-valued valuation $(\cdot)^w$ of λ-$\mathcal{CL}_m$ is an approximation (in the precision order) of the valuation $(\cdot)^{U,w}$ in the λ-canonical Kripke structure U. In other words, for any λ-world w and $\mathcal{CL}_m$ formula φ it is true that*

- *If $\varphi^w = \mathbf{f}$ then $U, w \not\models' \varphi$*
- *If $\varphi^w = \mathbf{t}$ then $U, w \models' \varphi$*

where U is the λ-canonical Kripke structure.

Proof. Take $w \in \mathcal{W}_\lambda$, U the λ-canonical Kripke structure and $\varphi \in \mathcal{COL}_m$. We will only prove that $\varphi^w = \mathbf{t}$ implies that $U, w \models' \varphi$, the other case then follows since $\varphi^w = \mathbf{f}$ iff $(\neg\varphi)^w = \mathbf{t}$ and since $U, w \not\models' \varphi$ iff $U, w \models' \neg\varphi$.

Assume $\varphi^w = \mathbf{t}$. It follows that for some $\alpha < \lambda : \varphi^{(w)_\alpha} = \mathbf{t}$. We prove $U, w \models' \varphi$ by induction on the structure of φ:

- If $\varphi = P(\bar{t})$, then

$$\begin{aligned}\mathbf{t} = P(\bar{t})^w &= lub_{\leq_p}\{P(t)^{(w)_\alpha} | \alpha < \lambda\} \\ &= lub_{\leq_p}\{P(t)^{((w)_\alpha)^{obj}} | \alpha < \lambda\} \\ &= P(t)^{((w)_0)}\end{aligned}$$

  So $\bar{t} \in P^{w^{obj}}$, hence $U, w \models' P(\bar{t})$.

- If $\varphi = \varphi_1 \wedge \varphi_2$, $\varphi = \neg\psi$ or $\forall x : \psi$ the proof is trivial.

- If $\varphi = K_A\psi$, there is an α such that $(K_A\psi)^{(w)_\alpha} = \mathbf{t}$, and by the monotonicity of the truth valuation: $(K_A\psi)^{(w)_{\alpha+1}} = \mathbf{t}$. Hence, for all $w' \in A^{(w)_{\alpha+1}}$: $\psi^{w'} = \mathbf{t}$ (by Proposition 6.2.6). It now follows that for all $w'' \in A^w : \psi^{w''} \geq_p \psi^{(w'')_\alpha} = \mathbf{t}$. Since $(w'')_\alpha \in A^{(w)_{\alpha+1}}$, this means that that $U, w'' \models' \psi$ for all $w'' \in \mathcal{K}_A$ and conclude that $U, w \models' K_A\psi$.

- If $\varphi = E_G\psi$, $\varphi^w = \mathbf{t}$ follows directly from the fact that $E_G\psi$ is syntactic sugar for $\bigwedge_{A\in\mathcal{A}} K_A\psi$ and the proof above.

- If $\varphi = C_G\psi$, we prove that $(C_G\psi)^w \neq \mathbf{u}$. Since $(C_G\psi)^w = glb_{\leq_t}\{(E^k_G\psi)^w | k \in \mathbb{N}_0\}$ it is sufficient to prove that $(E^k_G\psi)^w \neq \mathbf{u}$ for all $k \in \mathbb{N}_0$. This follows from the induction hypothesis since $C\psi^{R(w)} = \mathbf{t}$ implies that $(E^k_G\psi)^{R(w)} \neq \mathbf{u}$ for all $k \in \mathbb{N}_0$, which implies that $(E^k_G\psi)^w = \mathbf{t}$ (by induction hypothesis) for all k.

□

Proposition 6.2.9. *The formula $\varphi = O_A\neg Cp$ is not satisfiable in a ω-canonical Kripke structure.*

To prove this proposition, we introduce a graphical tree-representation of k-worlds ($k \in \mathbb{N}$).

Definition 6.2.10. We define a graph $Tree(w)$ for every k-world:

- For a 0-world: $Tree(w)$ is a node with label w^{obj}
- For a k-world: $Tree(w)$ is a graph with the node with label w^{obj} as root and an edge to a node with label A for every agent $A \in \mathcal{A}$ and an edge from every newly created node with label A to $Tree(w')$, for every $w' \in A^w$.

We say a node representing A ($A \in \mathcal{A}$) is an agent node and a node representing a structure is a non-agent node.

It is clear that this mapping is an isomorphism.

Proof. Assume for simplicity in notation a situation where we have one agent $\mathcal{A} = \{A\}$ and a vocabulary $\Sigma = \{p\}$. The proof does not depend on this simplification.

We will prove that given the above valuation, the formula

$$\neg O_A\neg Cp$$

will be true in U, w with U the ω-canonical Kripke Structure, and w any ω-world. So, towards contradiction, take a ω-world w such that $U, w \models O_A \neg Cp$. Then, by Proposition 6.2.6:

$$A^w = \{w' | U, w' \models' \neg Cp\}$$

By Definition 6.2.5, we know that for every $w' \in \mathcal{W}_\omega$:

$$U, w' \models' \neg Cp \text{ iff for some } k \in \mathbb{N} : U, w' \models \neg K_A^{k+1} p$$

By Proposition 6.2.6

$$(\neg K_A^{k+1} p)^{2v:(w')_k} = \mathbf{t}$$

Consider the class of worlds $\{w_k | k \in \mathbb{N}\}$, where w_k is a k-world such that $Tree(w_k)$ is a tree with a p in the root and in every node on the first k levels (see Figure 6.4). Each w_k satisfies $\bigwedge_{i\leq k} K_A^i p$. And consider the class of worlds $\{w'_{k+1} | k \in \mathbb{N}\}$, where w'_{k+1} is a $(k+1)$-world such that $Tree(w_k)$ is a tree with a p in the root and in every node on the first k levels and $\neg p$ on level $(k+1)$ (see Figure 6.5). Each w'_{k+1} satisfies $\bigwedge_{i\leq k} K_A^i p$ but falsifies $K_A^{k+1} p$

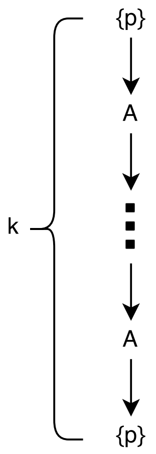

Figure 6.4: World w_k

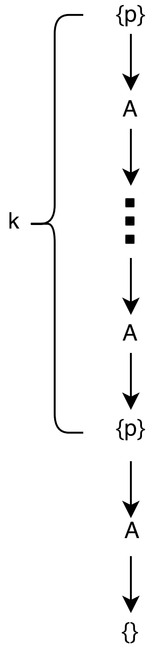

Figure 6.5: World w'_{k+1}

Now consider for each $k \in \mathbb{N}$ the set $[A^w]_k = \{(w')_k | w' \in A^w\}$. Since w'_{k+1} satisfies $\neg Cp$, it follows that $w'_{k+1} \in [A^w]_{k+1}$. Hence, $w_k = R(w'_{k+1}) \in$

$R[[A^w]_{k+1}] = [A^w]_{k+1}$. But then the sequence $\langle w_k \rangle_{k \in \mathbb{N}}$ of refining precision belongs to A^w. However, clearly $\langle w_k \rangle_{k \in \mathbb{N}}$ satisfies Cp.

Contradiction.

□

This is inherent to the approach with Kripke structures, with the accompanying two-valued valuation function. Since we have to fix the set of worlds beforehand we can only valuate formulas relative to the chosen set of worlds. This is in contrast to the valuation that we will propose. The three-valued valuation acknowledges when it "does not know" something.

A semantical structure in Belle and Lakemeyer, 2015

The key idea from the work in [Belle and Lakemeyer, 2015] was to propose infinite sequences of structures, where each member of a sequence represents the beliefs for a particular agent is compatible with the beliefs of its predecessor and extends it to account for one additional level of nested beliefs, as an extension to worlds with limited depth (as our k-worlds) they proposed in [Belle and Lakemeyer, 2010]. This is an approach similar to our ω-worlds, with some technical differences. This approach does however suffer from similar issues as shown in the last paragraph, as we will show in the example at the end of this paragraph. First, we will introduce their structures and semantics for $\mathcal{COL}_m$. In between the definitions we will discuss the connection with our k-worlds, ω-worlds, epistemic states A^w, ...

In this section, we will denote $\mathbb{W}$ as the set of all possible objective worlds, where a world is a set of ground atoms, as the authors defined it in [Belle and Lakemeyer, 2015]. The set of agents will be denoted by $I = \{1, \ldots, n\}$ and is fixed. We will also assume a domain D to be fixed throughout this section. We first describe k-structures, an account for the beliefs of an agent.

Definition 6.2.11. Let $k > 0$ and $n > 0$ the number of agents. Then define ξ^k as follows:

- $\xi^1 = 2^{\mathbb{W}}$
- $\xi^k = 2^{\mathbb{W} \times (\xi^{k-1})^n}$

An element of ξ^k is referred to as a k-structure.

A 1-structure is of the form $\{w', w'', \ldots\}$ and expresses what an agent believes about the world but has no account on the views of that agent about the beliefs of others. A 2-structure is of the form $\{(w', e'_1, e'_2, \ldots), (w'', e''_1, e''_2, \ldots), \ldots\}$ and expresses for example that the agent beliefs w' to be a possible state of the worlds, wherein he deems e'_1 to be a possible belief for agent 1.

To relate with our constructs:

- *A* 1*-structure* $\{w, w', w'', \ldots\}$ *is a set of objective structures, or* 0*-worlds. Or put differently: a 1-structure e is the same as* A^w *for* A *an agent and* w *a* 1*-world.*
- *A* 2*-structure* $\{(w, e_1, e_2, \ldots), (w', e'_1, e'_2, \ldots), \ldots\}$ *is a set of objective structures, with for each objective structure, for each agent a* 1*-structure (= a set of 0-worlds). This is the same as a set of* 1*-worlds, or the epistemic state* A^w *for an agent in a* 2*-worlds.*
- *In general, a* k*-structure is a set of* $(k-1)$*-worlds, or a epistemic state* A^w *of some agent* A *in a* k*-world.*

An *epistemic state* is used to determine the beliefs of all agents at all levels.

Definition 6.2.12. An *epistemic state* f is a function of the form $I \times \mathbb{N} \to \cup_{k=1}^{\infty} \xi^k$ such that for any i and $l \geq 1$: $f(i, l) \in \xi^k$.

Such an epistemic state is reasonable only when an agent's beliefs are consistent across all levels. That is, each level extends the previous level by adding another layer of (nested) beliefs about other agents' beliefs, while keeping the set of worlds an agent considers the same. They formalized this reasonability by defining *i-compatibility* between k-structures, and *proper* epistemic states. It is in these notions that the authors embedded assumptions about introspection and knowledge, which we chose not to do.

Definition 6.2.13. An epistemic state $e \in \xi^k$ and $e' \in \xi^{k+1}$ are i-compatible if:

- for $k = 1$:
$$e = \{w | (w, e_1, \ldots, e_i, \ldots, e_n) \in e'\} \text{ and } e_i = e;$$
- for $k > 1$:
  - $\{w | (w, e_1, \ldots, e_i, \ldots, e_n) \in e'\} = \{w | (w, e'_1, \ldots, e'_i, \ldots, e'_n) \in e'\}$ and $e'_i = e$
  - for every $(w, e'_1, \ldots, e'_i, \ldots, e'_n) \in e'$ there is a $(w, e_1, \ldots, e_i, \ldots, e_n) \in e$ such that e'_j and e_j are j-compatible for all $j \neq i$.

Definition 6.2.14. An epistemic state f is *proper* if for all i, for all $k > 0$: $f(i,k)$ and $f(i,k+1)$ are i-compatible.

To relate with our constructs:

- *An epistemic state is a function that assigns a k-structure to an agent i and a level k. So we can view it as a sequence s such that each element $(s)_k$ in the sequence contains a k-structure for each agent ($\approx A^w$, for w a k-structure). This is very similar to our ω-worlds: a sequence s such that each element $(s)_k$ in the sequence constains a k-world. An epistemic state (as defined by Belle and Lakemeyer) is in fact the same thing, but is has no account for the objective world in all these elements.*

Before we can define their semantics, we need one more concept. A *progression* of an epistemic state for an agent i in world w gives the view of agent i from world w.

Definition 6.2.15. Given an epistemic state f, its progression wrt an agent i and a world w is defined as:

$$f_i^w = \{\text{proper } f | \text{for all } k > 0 : \text{if } f'(1,k) = e_1, \dots, f'(n,k) = e_n,$$
$$\text{then } (w, e_1, \dots, e_n) \in f(i, k+1)\}$$

To relate with our constructs:

- *A progression maps an epistemic state f, an agent i and a world w to a set of epistemic states, such that f' is in that set if f' represents the view of agent i in world w in epistemic state f. This is very similar to our definition of epistemic state for ω-worlds in Definition 6.2.2.*

Using this they define a semantics, by defining when a pair (f, w) is a model of a formula.

Definition 6.2.16. The semantics are defined inductively:

- $f, w \models p$ iff $p \in w$.
- $f, w \models \neg\varphi$ iff $f, w \not\models \varphi$
- $f, w \models \psi \wedge \varphi$ iff $f, w \models \psi$ and $f, w \models \varphi$.
- $f, w \models \forall x : \varphi$ iff $f, w \models \psi[x/d]$ for each $d \in D$.

- $f, w \vDash K_i\varphi$ iff for all $w' \in f(i,1)$, for all $f' \in f_i^{w'}$: $f', w' \vDash \varphi$.
- $f, w \vDash E\varphi$ iff $f, w \vDash K_i\psi$ for all $i \in I$.
- $f, w \vDash C\varphi$ iff $f, w \vDash E^k\psi$ for all $k \geq 1$.
- $f, w \vDash O_i\varphi$ iff for all w' and $f', w' \in f(i,1)$ and $f' \in f_i^{w'}$ **IFF** $f', w' \vDash \psi$.

Given these definition we will show a translation of the example in Proposition 6.2.9, that exhibits the same issue for the semantics in [Belle and Lakemeyer, 2015][2].

Example 6.2.17. Given the semantics defined above, the following formula can never be true:

$$O_i\neg Cp$$

for any agent i. Without loss of generalization, we will assume that agent to be 1.

To prove this, assume towards contradiction that there is a f such that

$$f \models O_1\neg Cp$$

Note that we omit the w in the semantics since the valuation of $O_1\neg Cp$ is independent of the objective world w. Now take f' such that $f' \models K_1p \wedge \neg Cp$. It is clear that this should exist. Since $f \models O_1\neg Cp$, it follows that for each $g, v : g, v \models \neg Cp$ iff $v \in f(1,1)$ and $g \in f_1^v$. And since $f' \models K_1p \wedge \neg Cp$ it follows that $f'w' \models \neg Cp$ for all $w \in \mathbb{W}$, and as such $w' \in f(1,1)$ and $f' \in f_1^{w'}$. Note that this is true for every $w \in W$, which means: $\mathbb{W} = f(1,1)$.

We assume f' and f to be proper epistemic states, so from the fact that $f' \in f_1^{w'}$, we can derive that: $f'(1,1) = f(1,1)$, or equivalently $f'(1,1) = \mathbb{W}$. Since $(1,1) \subset w|w \models p$ since $f' \models K_1p$, this leads to the conclusion that $\forall w \in \mathbb{W} : w \models p$, which is a contradiction.

6.2.2 λ-worlds for $\mathcal{COL}_m$

In this section we propose a semantics for $\mathcal{COL}_m$ with deeper worlds, that do allow for valuation of formulas such as $O_A\neg Cp$.

Definition 6.2.18. Given a $\varphi \in \mathcal{COL}_m$ and a μ-world w, we define a three-valued valuation function φ^w by structural induction with the rules of Definition

[2]I would like to thank Vaishak Belle for his assistance exploring this issue and for coming up with the proof in this example.

6.1.14, augmented with:

$$(O_A\varphi)^w = \begin{cases} \mathbf{u} & \text{if } w \text{ is a 0-world} \\ \mathbf{u} & \text{if } \exists w' \in \mathcal{W}_\mu \text{ such that } \varphi^{w'} = \mathbf{u} \\ \mathbf{t} & \text{if } A^w = \{w' \in \mathcal{W}^\mu | \varphi^{w'} = \mathbf{t}\} \\ \mathbf{f} & \text{otherwise} \end{cases}$$

These three-valued semantics are different from the semantics proposed in the intermezzo. Intuitively, they differ in the sense that they admit that sometimes, they do not know something. If a world is not deep enough to evaluate the formula in the scope of the O operator, the valuation of the O operator will by itself be unknown.

This valuation seems to satisfy all concerns expressed above. However, when studying this in more detail, we noticed that proposition 6.1.18 (and by consequence, all results afterwards) are not valid for $\mathcal{COL}_m$ under this semantics. Intuitively, the reason is that the O operator behaves completely different than the other operators. If something is known ($K_A\varphi$), then adding more information on a deeper level, does not make this "un-known". This is different when talking about what someone does not know (or by consequence, what someone only knows). We give an example of what might go wrong:

Example 6.2.19. Consider the 1-world w and the 2-world w' in Figures 6.6 and 6.7.

Clearly, it holds that $w' \geq_p w$. Yet it is easy to see that $(O_A(p \vee q))^w = \mathbf{t}$, while $(O_A(p \vee q))^{w'} = \mathbf{f}$. So this means that while $w' \geq_p w$, it is not necessarily the case that $\varphi^{w'} \geq_p \varphi^w$ for any $\mathcal{COL}_m$ formula φ.

Losing this property means that we can now have sequences of worlds of increasing precision, where the truth valuation of some formula is not increasing in precision. We can have a λ-world $w = \langle v_1, \ldots, v_n, v_{n+1}, \ldots \rangle$, where it is possible that a formula $O_A(\varphi)$ is true in v_n, but false in v_{n+1}. This means that we lose many important mathematical properties like Proposition 6.1.21. The valuation of a formula is not always defined in a λ-world, since the least upper bound of $\{\mathbf{t}, \mathbf{f}\}$ in the precision order is not defined in a three-valued setting.

In the next section we propose an alternative construct for the O_A operator that states how deep an agent "only knows" something. This construct is monotone for the truth valuation of the formulas. Using this, the goal is to propose an operator that can only be resolved in worlds from a certain (fixed) depth, but captures only knowing of all formulas that can be formed in $\mathcal{COL}_m$. This has been investigated in the context of Kripke-structures with the help of characteristic formulas [Aucher and Belle, 2015]. We want to generalize

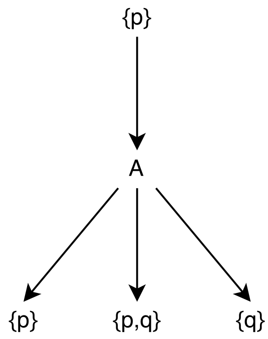

Figure 6.6: World w, or $R(w')$

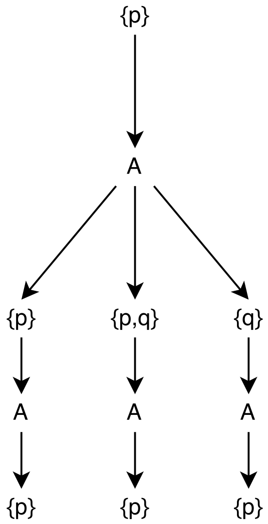

Figure 6.7: World w'

our semantics based on possible worlds, because of the shortcomings of a Kripke-based approach that we discussed in the introduction of this chapter.

6.3 Limited Only-knowing

In this thesis, we analyze the monotonicity issue above by replacing the only-knowing operator with a set of operators O^{α}_{A}, intuitively stating: "I only know ... up until level α" (with until level α, we mean α nested references to an agents knowledge).

Definition 6.3.1. Given a vocabulary Σ, an indexed set of agents $\mathcal{A}$ and an ordinal γ, we define the language $\mathcal{CO}^{\gamma}\mathcal{L}_m$ by structural induction with the rules of Definition 6.2.1, with the last rule replaced by:

$$O^{\alpha}_{A}(\psi) \qquad \text{is a formula if } \alpha < \gamma \text{ is an ordinal, } \psi \text{ is a formula and } A \in \mathcal{A}$$

Definition 6.3.2. Given an ordinal γ, a $\varphi \in \mathcal{CO}^{\gamma}\mathcal{L}_m$ and a μ-world w ($\mu = 0$ or μ a successor ordinal), we define a three-valued valuation function φ^{w} by structural induction with the rules of Definition 6.2.18, with the last rule replaced by:

$$(O_A^\alpha \varphi)^w = \begin{cases} \mathbf{u} & \text{if } \mu = 0 \text{ or } \mu < \alpha \\ \mathbf{u} & \text{if } \exists w' \in \mathcal{W}_{\mu-1} \text{ such that } \varphi^{w'} = \mathbf{u} \\ \mathbf{t} & \text{if } X_\alpha[A^w] = X_\alpha[\{w' \in \mathcal{W}^{\mu-1} | \varphi^{w'} = \mathbf{t}\}] \\ \mathbf{f} & \text{otherwise} \end{cases}$$

To illustrate, we look again at Example 6.2.19.

Example 6.3.3. In both w and w', we have $O_A^1(p \vee q)^w = (O_A^1(p \vee q))^{w'} = \mathbf{t}$, while $(O_A^2(p \vee q))^w = \mathbf{u}$, and $(O_A^2(p \vee q))^{w'} = \mathbf{f}$. For every $k > 2$: $O_A^k(p \vee q)^w = \mathbf{u}$, and $O_A^k(p \vee q)^{w'} = \mathbf{u}$. As such, for each α, we have in this example that $O_A^\alpha(p \vee q)^w \leq_p O_A^\alpha(p \vee q)^{w'}$.

Definition 6.3.4. Given an ordinal γ, the *modal depth $MD(\varphi)$* of a formula $\varphi \in \mathcal{CO}^\gamma\mathcal{L}_m$ is defined by induction on the structure of φ, as in definition 6.1.22, with the last rule replaced by:

$$MD(O_A^\alpha \varphi) = Max(MD(\varphi) + 1, \alpha)$$

Proposition 6.3.5. *Given an ordinal γ, any formula φ in $\mathcal{CO}^\gamma\mathcal{L}_m$, then $MD(\varphi) < MAX(\omega^2, \gamma\omega)$.*

Proof. If $\gamma < \omega$ then every operator heightens the modal depth of a formula by at most ω, while if $\gamma \geq \omega$: every operator heightens the modal depth of a formula by at most γ. Since a formula is of finite length, it has to be the case that $MD(\varphi) < MAX(\omega^2, \gamma\omega)$. $\square$

The propositions and theorems from Section 6.1.3 are true for $\mathcal{CO}^\gamma\mathcal{L}_m$, for any ordinal γ as the following propositions and theorems will show. The proofs are very similar, so we will shorten them and only give the relevant parts.

Proposition 6.3.6. *For any μ-world w, any ordinal γ, and any formula in $\mathcal{CO}^\gamma\mathcal{L}_m$:*

$$\varphi^w \geq_p \varphi^{R(w)}$$

Proof. We prove that for any arbitrary $(\mu + 1)$-world w and any formula $\varphi \in \mathcal{CO}^\gamma\mathcal{L}_m$: $\varphi^w \geq_p \varphi^{R(w)}$. For the same reasons as before, we only prove that $\varphi^w = \mathbf{t}$ follows from $\varphi^{R(w)} = \mathbf{t}$. We prove this by induction on the structure of φ.

- If $\varphi = P(\bar{t})$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi = \neg\psi$, $\varphi = \forall x\psi$, $\ldots$, $K_A\psi$, $E_G\psi$, $C\psi$ the proof is the same as for Proposition 6.1.18.

- If $\varphi = O^{\alpha}_{A}\psi$ then
  - We know that $\mu + 1 > \mu > \alpha$, since $\varphi^{R(w)} \neq \mathbf{u}$.
  - Assume there is a $w' \in \mathcal{W}_{\mu-1}$ such that $\psi^{w'} = u$. Then $\psi^{R(w')} = u$ following from contraposition of the induction hypothesis, and by consequence: $\varphi^{R(w)} = u$.
  - To prove that $(O^{\alpha}_{A}\psi)^{w} = \mathbf{t}$, we prove

    $$X_{\alpha}[A^{w}] = X_{\alpha}[\{w' \in \mathcal{W}^{\mu-1} | \psi^{w'} = \mathbf{t}\}]$$

    From $\varphi^{R(w)} = (O^{\alpha}_{A}\psi)^{R(w)} = \mathbf{t}$, we derive:

    $$X_{\alpha}([A^{R(w)}] = X_{\alpha}[\{w'' \in \mathcal{W}^{\mu-2} | \psi^{w''} = \mathbf{t}\}]$$

    Since $A^{R(w)} = R[A^{w}]$ and $\alpha < \mu - 1$, we find:

    $$\begin{aligned} X_{\alpha}[A^{w}] = X_{\alpha}[A^{R(w)}] &= X_{\alpha}[\{w'' \in \mathcal{W}^{\mu-2} | \psi^{w''} = \mathbf{t}\}] \\ &= X_{\alpha}[\{w'' | w'' \in R[\mathcal{W}^{\mu-1}] \wedge \psi^{w''} = \mathbf{t}\}] \\ &= X_{\alpha}[\{R(w'') | w'' \in \mathcal{W}^{\mu-1} \wedge \psi^{w''} = \mathbf{t}\}] \end{aligned}$$

    Using the induction hypothesis, we find:

    $$X_{\alpha}[A^{w}] = X_{\alpha}[\{w'' | w'' \in \mathcal{W}^{\mu-1} \wedge \psi^{w''} = \mathbf{t}\}]$$

    which was to be proven.

This proves that $\varphi^{w} = \mathbf{t}$ for all $\varphi \in \mathcal{CO}^{\gamma}\mathcal{L}_{m}$. □

Proposition 6.3.7. *For any ordinal γ, any μ-world w and μ'-world w' such that $w' \geq_{p} w$: for any formula φ in $\mathcal{CO}^{\gamma}\mathcal{L}_{m}$: $\varphi^{w'} \geq_{p} \varphi^{w}$*

Proof. The proof is the same as the proof for Proposition 6.1.21. □

Theorem 6.3.8. *Given an ordinal γ, a μ-world w (μ an arbitrary ordinal) can resolve every formula $\varphi \in \mathcal{CO}^{\gamma}\mathcal{L}_{m}$, with $MD(\varphi) \leq \mu$, i.e., φ^{w} is 2-valued for every μ-world w.*

Proof. We prove that a μ-world resolves all $\mathcal{CO}^{\gamma}\mathcal{L}_{m}$ formulas with modal depth $\leq \mu$.

First, assume that μ is zero, or is a successor ordinal. Take a μ-world w, and a formula $\varphi \in \mathcal{CO}^{\gamma}\mathcal{L}_{m}$, with $MD(\varphi) \leq \mu$. We prove that $\varphi^{w} \neq \mathbf{u}$ by induction on the structure of φ.

- If $\varphi = P(\bar{t})$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi = \neg\psi$, $\varphi = \forall x\psi$, $\ldots$, $K_A\psi$, $E_G\psi$, $C\psi$ the proof is the same as for Proposition 6.1.25.
- If $\varphi = O_A^\alpha\psi$, to prove that $\varphi^w \neq \mathbf{u}$, we need to asses two cases:
  - $\varphi^w = \mathbf{u}$ if $\mu < \alpha$, but we know that $\mu \geq MD(\varphi) = max(MD(\psi) + 1, \alpha) \geq \alpha$.
  - $\varphi^w = \mathbf{u}$ if there is a $(\mu - 1)-$world w' such that $\psi^{w'} = \mathbf{u}$, which cannot be, as we can derive from the induction hypothesis.

Now, assume μ is a limit ordinal. Take a μ-world w, and a formula $\varphi \in \mathcal{CO}^\gamma\mathcal{L}_m$, with $MD(\varphi) < \mu$. We know that $MD(\varphi) < \mu$ and μ is a limit ordinal. Define $\alpha = MD(\varphi) + 1$, then $\alpha < \mu$. Using the induction hypothesis, we find that $\varphi^{(w)_\alpha} \neq \mathbf{u}$, and from the monotonicity of the valuation (Proposition 6.3.7) it follows that

$$\varphi^w = lub_{\leq_t}\{\varphi^{(w)_\alpha} | \alpha < \mu\} \neq \mathbf{u}$$

This proves that $\varphi^w \neq \mathbf{u}$ for all $\varphi \in \mathcal{CO}^\gamma\mathcal{L}_m$. And by Proposition 6.3.7, it follows that $\varphi^{w'} \geq_p \varphi^w$. □

We define for every ordinal μ the logic μ-$\mathcal{CO}^\gamma\mathcal{L}_m$ as the logic with language $\mathcal{CO}^\gamma\mathcal{L}_m$ and the valuation as defined above, using μ-worlds.

Theorem 6.3.9. *For any ordinal γ, μ-$\mathcal{CO}^\gamma\mathcal{L}_m$ has a (not necessarily strict) larger set of satisfiable formulas than μ'-$\mathcal{CO}^\gamma\mathcal{L}_m$ if $\mu > \mu'$.*

Proof. Similar to the proof of Theorem 6.1.26. □

Consequence 6.3.1. *For any ordinal $\gamma < \omega^2$ it is the case that for any ω^2-world w the valuation $(\cdot)^w$ is 2-valued in $\mathcal{CO}^\gamma\mathcal{L}_m$.*

Proof. Take $\varphi \in \mathcal{CO}^\gamma\mathcal{L}_m$ and $w \in \mathcal{W}_{\omega^2}$. Since $\gamma < \omega^2$ there is a k such that $\gamma < k\omega$. Now, by Proposition 6.3.5, we know that $MD(\varphi) < MAX(\omega^2, k\omega) = \omega^2$. So, using Theorem 6.3.8: $\varphi^w \neq \mathbf{u}$. □

Consequence 6.3.2. *For any $\alpha < \omega^2$, there is no α-$\mathcal{CO}^\gamma\mathcal{L}_m$ logic with a strict larger set of satisfiable formulas than ω^2-$\mathcal{CO}^\gamma\mathcal{L}_m$.*

Proof. Similar to the proof of Theorem 6.1.3. □

Consequence 6.3.2 is an interesting result, as it fixes an ordinal μ for each γ such that μ-$\mathcal{CO}^\gamma\mathcal{L}_m$ has a maximal set of satisfiable formulas. There is however still some work left open. Naturally, we cannot find an μ such that μ-$\mathcal{CO}^\gamma\mathcal{L}_m$ has a maximal set of satisfiable formulas, for any γ. We would however like a result where we can define an O_A^α operator, with a fixed α (and use O_A as a shorthand for this operator), since choosing an α to explicitly write how deep the only knowing is, is not natural. This is a technical aspect, not relevant for a knowledge engineer.

A good choice for fixing α seems to be $\alpha = \omega^2$. This choice is motivated by the fact there is nothing that can be expressed in $\mathcal{CL}_m$ that has a modal depth $> \omega^2$. Ideally, we would like to fix α to an ordinal γ such that γ-$\mathcal{CO}^\gamma\mathcal{L}_m$ has a maximal set of satisfiable formulas.

We have not found this γ. An hypothesis is that this γ does not exist, since fixing $\gamma < \omega^2$ allows for writing formulas $\varphi = C^k\psi$ such that $MD(\varphi) \geq k\omega\gamma$ (and probably $\varphi^w = \mathbf{u}$ for a $w \in \mathcal{W}^\mu$), while fixing $\mu \geq \omega^2$ allows for writing formulas $\varphi = O_AO_A\psi$ such that $MD(\varphi) = 2\mu > \mu$.

A partial solution we propose is to limit nesting of O operators. If we do not allow nesting of O operators, we can fix α to ω^2, since the only formulas that can occur within a O operator are of depth $< \omega^2$. This means it makes sense to fix α to ω^2, since $\mathcal{CL}_m$ formulas are the only kind of statement an agent can or cannot know.

Definition 6.3.10. Given a vocabulary Σ and an indexed set of agents $\mathcal{A}$, we define the language $\mathcal{CO}^{\omega^2}\mathcal{L}_m$ by structural induction with the rules of Definition 6.2.1, with the last rule replaced by:

$$O_A(\psi) \qquad \text{is a formula if } \psi \text{ is a formula in } \mathcal{CL}_m \text{ and } A \in \mathcal{A}$$

Definition 6.3.11. Given a $\varphi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m$ and a μ-world w, we define a three-valued valuation function φ^w by structural induction with the rules of Definition 6.2.18, with the last rule replaced by:

$$(O_A\varphi)^w = \begin{cases} \mathbf{u} & \text{if } \mu < \omega^2 \\ \mathbf{u} & \text{if } \exists w' \in \mathcal{W}_{\mu-1} \text{ such that } \varphi^{w'} = u \\ \mathbf{t} & \text{if } X_{\omega^2}[A^w] = X_{\omega^2}[\{w' \in \mathcal{W}^{\omega^2} | \varphi^{w'} = t\}] \\ \mathbf{f} & \text{otherwise} \end{cases}$$

Definition 6.3.12. The *modal depth* $MD(\varphi)$ of a formula $\varphi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m$ is defined by induction on the structure of φ, as in definition 6.1.22, augmented by:

$$MD(O_A\varphi) = Max(MD(\varphi) + 1, \omega^2)$$

Following propositions, theorems and consequences follow directly from the corresponding propositions, theorems and consequences of $\mathcal{CO}^{\gamma}\mathcal{L}_m$.

Proposition 6.3.13. *Given any formula φ in $\mathcal{CO}^{\omega^2}\mathcal{L}_m$, then $MD(\varphi) < 2\omega^2$.*

Consequence 6.3.3. *For any $2\omega^2$-world w the valuation $(\cdot)^w$ is two valued in $\mathcal{CO}^{\omega^2}\mathcal{L}_m$.*

Consequence 6.3.4. *There is no α-$\mathcal{CO}^{\omega^2}\mathcal{L}_m$ logic with a strict larger set of satisfiable formulas than $2\omega^2$-$\mathcal{CO}^{\gamma}\mathcal{L}_m$.*

In [Aucher and Belle, 2015], the authors also use a similar limited only-knowing construction. They have also defined a O_A operator without a index, stating the depth. But different to our goal here, this operator does not capture lack of knowledge of arbitrary depth. It is equivalent to our O_A^0 operator.

6.4 Other extensions

6.4.1 $\mathcal{CO}^{\omega^2}\mathcal{L}_m$with positive introspection, negative introspection and/or (truthful) knowledge

As stated above, $\mathcal{CO}^{\omega^2}\mathcal{L}_m$ does not satisfy many modal axioms by default. However, this modification can easily be formulated, by simply requiring it and defining the semantics over the appropriate subset. First we define what it means to be introspective and what it means to be truthful (or informed).

Definition 6.4.1. We define a $\mu + 1$-world w ($\mu > 1$) to be **positive introspective** if for all $A \in \mathcal{A}$ and for each $w' \in A^w$:

$$A^{w'} \subseteq R[A^w]$$

We define it to be **negative introspective** if for all $A \in \mathcal{A}$ and for each $w' \in A^w$:

$$R[A^w] \subseteq A^{w'}$$

We define it to be **informed** if for all $A \in \mathcal{A}$:

$$R(w) \in A^w$$

Definition 6.4.2. We define a λ-world w to be **positive introspective** if for all $A \in \mathcal{A}$ and for each $w' \in A^w$:

$$A^{w'} \subseteq A^w$$

We define it to be **negative introspective** if for all $A \in \mathcal{A}$ and for each $w' \in A^w$:

$$A^w \subseteq A^{w'}$$

We define it to be **informed** if for all $A \in \mathcal{A}$:

$$w \in A^w$$

We denote the subset of all positive introspective, negative introspective and informed λ-worlds as $\mathcal{W}_\lambda^{pi}$, $\mathcal{W}_\lambda^{ni}$, respectively $\mathcal{W}_\lambda^{in}$ and will also use shorthands as $\mathcal{W}_\lambda^{pi,in}$ for (for example) positive introspective informed worlds. We now formally prove that the logics satisfy the relevant axioms.

Theorem 6.4.3. *Given a μ-world w ($\mu \geq 2$), an agent A and arbitrary formula $\varphi, \psi \in \mathcal{CO}^\gamma\mathcal{L}_m$, following statements hold:*

1. *$w \models K_A\varphi \wedge K_A(\varphi \Rightarrow \psi) \Rightarrow K_A\psi$ (Rational agents)*
2. *If $w \in \mathcal{W}_\mu^{pi}$: $w \models K_A\varphi \Rightarrow K_AK_A\varphi$ (Positive Introspection)*
3. *If $w \in \mathcal{W}_\mu^{ni}$: $w \models \neg K_A\varphi \Rightarrow K_A\neg K_A\varphi$ (Negative Introspection)*
4. *If $w \in \mathcal{W}_\mu^{in}$: $w \models K_A\varphi \Rightarrow \varphi$ (Knowledgeable)*

Proof. We prove this by induction on μ. Assume μ is a successor ordinal.

1. Take $w \in \mathcal{W}_\mu$ and $\varphi, \psi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m$ such that $w \models K_A\varphi \wedge K_A(\varphi \Rightarrow \psi)$ arbitrarily. If $w \vDash K_A\varphi$, then $w' \vDash \varphi$ for all $w' \in A^w$. But since $w \vDash K_A(\varphi \Rightarrow \psi)$, we also have that $w' \vDash \psi$. By consequence $w' \vDash \psi$ for all $w' \in A^w$, or equivalently: $w \vDash K_A\psi$.
2. Take $w \in \mathcal{W}_\mu^{pi}$ and $\varphi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m$ arbitrarily. If $w \vDash K_A\varphi$, then $w' \vDash \varphi$ for all $w' \in A^w$. But since $A^{w'} \subseteq R[A^w]$, we also have that $w'' \vDash \varphi$ for all $w'' \in A^{w'}$. Since w' was arbitrarily chosen, we can conclude that $w \vDash K_AK_A\varphi$.
3. Similar to 2.
4. Similar to 2.

Now, assume μ is a limit ordinal.

1. Take $w \in \mathcal{W}_\mu$, $w' \in A^w$ and $\varphi, \psi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m$ arbitrarily. Since

$$(w)_\alpha \models K_A\varphi \wedge K_A(\varphi \Rightarrow \psi) \Rightarrow K_A\psi$$

for each $\alpha < \mu$ by induction and

$$(K_A\varphi \wedge K_A(\varphi \Rightarrow \psi) \Rightarrow K_A\psi)^w =$$

$$lub_{\leq_p}\{(K_A\varphi \wedge K_A(\varphi \Rightarrow \psi) \Rightarrow K_A\psi)^{(w)_\alpha} | \alpha < \lambda\}$$

it follows that $w \models (K_A\varphi \wedge K_A(\varphi \Rightarrow \psi) \Rightarrow K_A\psi)$.

2. Take $w \in \mathcal{W}_\mu$, $w' \in A^w$ and $\varphi, \psi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m$ arbitrarily. From $A^{w'} \subseteq A^w$ it follows that $A^{(w')_\alpha} \subseteq A^{(w)_\alpha}$ for each $\alpha < \lambda$. This means that for each $\alpha < \lambda$: $(w)_\alpha \in \mathcal{W}^{pi}_\alpha$ and we can use the induction hypothesis to find that

$$(K_A\varphi \Rightarrow K_AK_A\varphi)^w =$$

$$lub_{\leq_p}\{(K_A\varphi \Rightarrow K_AK_A\varphi)^{(w)_\alpha} | \alpha < \lambda\} = \mathbf{t}$$

3. Similar to 2.

4. Similar to 2.

□

6.4.2 Public Announcements

Public announcements are modal propositions that produce common knowledge of the announced proposition. In many epistemic riddles, public announcements are essential because, as shown by the Coordinated attack problem [Gmytrasiewicz and Durfee, 1992], it is the only way to establish common knowledge amongst a group of agents. Intuitively, $w \vDash [\varphi]\psi$ means ψ will be true in the world $w[\varphi]$ obtained by publicly announcing φ in world w. The effect of this is that for all agents A and worlds $w' \in A^w$, w' is deleted from A^w if φ is not knowledge in w'. Every agent adjusts his belief and filters out the worlds that are not consistent with φ. Since every agent also knows that other agents have heard this announcement, he then recursively also filters the knowledge he has on others. This filtering is recursively applied all the way down.

Syntactically, we define a language $\mathcal{CO}^{\omega^2}\mathcal{L}^+_m$, by adding an operator to the syntax of $\mathcal{CO}^{\omega^2}\mathcal{L}_m$: $[\varphi]\psi$: ψ is true after φ has been publicly announced.

Definition 6.4.4. Given a vocabulary Σ and an indexed set of agents $\mathcal{A}$, we define the language $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$ by structural induction with the rules of Definition 6.3.1, augmented with:

$$[\varphi]\psi \qquad \text{is a formula if } \varphi, \psi \text{ are formulas}$$

To define the semantics, we will first define a function *Confine* that filters epistemic states as discussed above. Given a set of worlds M and a world w, it maps to a set of worlds W, where in every world, every agent's belief is consistent with M. With this function we can define the world $w[\varphi]$ obtained by publicly announcing φ in w.

Definition 6.4.5. We define for each $\mu \geq 0$ a function *Confine* :

$$\mathcal{W}_\mu \times 2^{\mathcal{W}_\mu} \to \mathcal{W}_\mu$$

that maps a world w and a set of worlds M to a world $\mathit{Confine}_\mu(w, M)$ such that $\mathit{Confine}_\mu(w, M)$ is a singleton containing the world obtained by publicly announcing in world w that all worlds outside M are impossible, or the empty set if it is not possible in w that all worlds outside of M are impossible. We define this by induction on μ:

- If $\mu = 0$:
$$W = \{w\} \cap M$$
- If $\mu = \mu + 1$:
$$W = M \cap \{(w^{obj}, (N_A)_{A \in \mathcal{A}})\}$$
where
$$N_A = \bigcup_{v \in A^w} \mathit{Confine}_\mu(v, R[M])$$
- If μ is a limit ordinal,
$$\mathit{Confine}(w, M) = M \cap \{\langle v_\alpha \rangle_{\alpha < \mu} | v_\alpha \in \mathit{Confine}_\alpha((w)_\alpha, X_\alpha[M])\}$$

We define *Confine* as the union of all $\mathit{Confine}_\mu$ functions.

Definition 6.4.6. We define for each μ world w and formula φ such that $w \models \varphi$ a world

$$w[\varphi] = \mathit{Confine}(w, \{w' | w' \in \mathcal{W}_\mu \text{ and } w' \vDash \varphi\})$$

We need to prove that $w[\varphi]$ is a consistent world, to show that it is well-defined.

Proposition 6.4.7. *If w is a consistent μ-world, and M a set of consistent μ-worlds, then all worlds in Confine(w, M) are consistent.*

Proof. We proof this by induction on μ.

- Assume $\mu = 0$, $w \in \mathcal{W}_0$ and $M \subseteq \mathcal{W}_0$. Since $\mathit{Confine}(w, M) = \{w\} \cap M$, this can only contain consistent worlds by definition.
- Assume $\mu = \mu' + 1$, $w \in \mathcal{W}_\mu$ and $M \subseteq \mathcal{W}_\mu$, then $\mathit{Confine}(w, M) = M \cap \{(w^{obj}, (N_A)_{A\in\mathcal{A}})\}$. Since M is a set of consistent worlds, we need to prove that $(w^{obj}, (N_A)_{A\in\mathcal{A}})$ is a consistent world. By definition of consistency, this is the case if for all A: N_A is a set of consistent worlds, which follows from the induction hypothesis.
- Assume $\mu = \lambda$, $w \in \mathcal{W}_\lambda$ and $M \subseteq \mathcal{W}_\lambda$. If $\mathit{Confine}(w, M)$ is empty, the proposition is trivially satisfied, so assume it is not empty. Then it is a singleton, by construction, so take w' the only element in $\mathit{Confine}(w, M)$. Since $\mathit{Confine}(w, M)$ is not empty, it follows that $\mathit{Confine}_\alpha((w)_\alpha, X_\alpha[M])$ is not empty for all $\alpha < \lambda$. Abusing notation, we will write $\mathit{Confine}_\alpha((w)_\alpha, X_\alpha[M])$ to denote the only element in that set. Then $w' = \langle v_\alpha\rangle_{\alpha<\mu}$, with $v_\alpha = \mathit{Confine}((w)_\alpha, ([M])_\alpha)$. w' is consistent
  - if $(w')_\alpha$ is consistent for all $\alpha < \lambda$, and
  - if for each $\alpha' < \alpha'' < \lambda$: $(w')_{\alpha'} \leq_p (w')_{\alpha''}$.

  Since $(w')_\alpha$ is consistent for all $\alpha < \lambda$, we only need to prove the second. We only prove that
  $$R(v_{\alpha+1}) = v_\alpha$$
  for an arbitrary $\alpha < \lambda$. The fact that for each $\alpha < \alpha' < \lambda$: $v_\alpha \leq_p v_{\alpha'}$ then follows easily by induction as in the proofs before. By Lemma 6.4.8 (below):
  $$\begin{aligned} R(v_{\alpha+1}) &= R(\mathit{Confine}((w)_{\alpha+1}, ([M])_{\alpha+1})) \\ &= \mathit{Confine}(R((w)_{\alpha+1}), R(([M])_{\alpha+1})) \\ &= \mathit{Confine}((w)_\alpha, ([M])_\alpha) \\ &= (w)_\alpha \end{aligned}$$
  This proves that $\mathit{Confine}(w, M)$ is consistent.

□

Lemma 6.4.8. *For a $(\mu+1)$-world w and a set of $(\mu+1)$-worlds M where $\mathit{Confine}(R(w), R[M]) \neq \emptyset$, then $\mathit{Confine}(w, M) \neq \emptyset$ and:*

$$R(\mathit{Confine}(w, M)) = \mathit{Confine}(R(w), R[M])$$

Proof. We define this by induction on μ:

- Assume $\mu = 0$, $w \in \mathcal{W}_1$ and $M \subseteq \mathcal{W}_1$. Then:

$$\begin{aligned}\{R(w)\} \cap R[M] &= R[\{w\}] \cap R[M] \\ &= R[\{w\} \cap M] \\ &= R(\mathit{Confine}(w, M))\end{aligned}$$

- Assume $\mu = \alpha + 1$, $w \in \mathcal{W}_{\alpha+1}$ and $M \subseteq \mathcal{W}_{\alpha+1}$. Then define $W_1 = \mathit{Confine}(R(w), R[M])$ and $W_2 = R(\mathit{Confine}(w, M))$. By assumption $W_1 \neq \emptyset$, so take $w' \in W_1$. Then $w' = (w^{obj}, (A^{w'})_{A \in \mathcal{A}})$, with

$$A^{w'} = \cup_{v \in A^{R(w)}} \mathit{Confine}(v, R[R[M]])$$

Since $A^{R(w)} = R[A^w]$, it follows by induction hypothesis that

$$\begin{aligned}A^{w'} &= \cup_{v \in A^{R(w)}} \mathit{Confine}(v, R[R[M]]) \\ &= \cup_{v \in R[A^w]} \mathit{Confine}(v, R[R[M]]) \\ &= \cup_{v \in A^w} \mathit{Confine}(R(v), R[R[M]]) \\ &= \cup_{v \in A^w} R[\mathit{Confine}(v, R[M])]\end{aligned}$$

So $w' \in W_2$.

- The proof when $\mu = \lambda$ is similar.

□

Using this function, we define the semantics for $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$:

Definition 6.4.9. For every $\mu + 1$-world w and $\varphi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m^+$, we define that w satisfies φ (notation $w \models \varphi$) by the structural induction from definition 6.3.2, augmented with one rule:

$$([\varphi]\psi)^w = \begin{cases} \mathbf{u} & \text{if } \varphi^w = \mathbf{u} \\ \mathbf{t} & \text{if } \varphi^w = \mathbf{f} \\ \psi^{w[\varphi]} & \text{otherwise} \end{cases}$$

This operator has the same reasonable properties from [Lutz, 2006] as $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$ as defined in [Belle and Lakemeyer, 2015].

Proposition 6.4.10. *Following properties are true for $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$, with worlds $w \in \mathcal{W}_\mu$ with $\mu > MD(\varphi)$:*

- $[\varphi]p \Leftrightarrow (\varphi \Rightarrow p)$ *is valid*
- $[\varphi]\neg\alpha \Leftrightarrow (\varphi \Rightarrow \neg[\varphi]\alpha)$ *is valid*
- $[\varphi]\alpha \wedge \beta \Leftrightarrow ([\varphi]\alpha \wedge [\varphi]\beta)$ *is valid*
- $[\varphi]K_A\alpha \Leftrightarrow (\varphi \Rightarrow K_A(\varphi \Rightarrow [\varphi]\alpha))$ *is valid*
- $[\varphi]\forall x\alpha \Leftrightarrow \forall x([\varphi]\alpha)$ *is valid*

Proof. We will only prove the first and fourth item, since the others can be proven by induction on the structure of the formula in a similar way.

Take w an arbitrary μ-world, we prove that $w \models [\varphi]p$ if and only if $w \models (\varphi \Rightarrow p)$. We distinguish two cases:

- If $w \not\models \varphi$, then trivially satisfies both $[\varphi]p$ and $(\varphi \Rightarrow p)$.
- If $w \models \varphi$, we prove that $w \models p$ implies that $w[\varphi] \models p$ and vice versa. We know that $w \models [\varphi]p$ if and only if $w' \models p$, with[3]

  $$w' = \mathit{Confine}(w, \{v | v \in \mathcal{W}_\mu \text{ and } v \models \varphi\})$$

  $$= (w^{obj}, (A^{w'})_{A \in \mathcal{A}}\})$$

  and $(p)^w = p^{w^{obj}} = p^{w'^{obj}}$. So this proves that $w \models p$ if and only if $w[\varphi] \models p$.

This proves that $w \models \varphi \rightarrow p$ if and only if $w[\varphi] \models p$.

Take w an arbitrary μ-world, we prove that $w \models [\varphi]K_Bp$ if and only if $w \models \varphi \Rightarrow K_B(\varphi \Rightarrow [\varphi]p)$. The fact that $w \models [\varphi]K_A\alpha \Leftrightarrow (\varphi \Rightarrow K_A(\varphi \Rightarrow [\varphi]\alpha))$ for arbitrary formulas α can then be proven by straightforward induction on the structure of α. As before, we can safely assume that $w \models \varphi$. We know that $w \models [\varphi]K_Bp$ if and only if $w' \models K_Bp$, with

$$w' = \mathit{Confine}(w, \{v | v \in \mathcal{W}_\mu \text{ and } v \models \varphi\})$$

$$= (w^{obj}, (A^{w'})_{A \in \mathcal{A}}\})$$

[3]Since $w \models \varphi$, $\mathit{Confine}(w, \{v | v \in \mathcal{W}_\mu$ and $v \models \varphi\})$ is not empty and can be identified with a world w', the only element in that singleton.

where $A^{w'} = \bigcup_{v \in A^w} \mathit{Confine}(v, R[M])$. We know that $w' \models K_B p$ iff for each $v' \in B^{w'} = \bigcup_{v \in A^w} \mathit{Confine}(v, R[M])$: $v' \models p$. So, stated differently: $w \models [\varphi]K_B p$ if for each $v \in A^w$: $\mathit{Confine}(v, R[M]) \models p$, or $v \not\models \varphi$. This is equivalent with v satisfying $\varphi \Rightarrow [\varphi]p$, as we have proven before. $\square$

The propositions and theorems from Section 6.1.3 are true for $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$, as the following propositions and theorems will show. The proofs are very similar, so we will shorten them and only give the relevant parts.

Proposition 6.4.11. *For any μ-world w and any formula in $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$:*

$$\varphi^w \geq_p \varphi^{R(w)}$$

Proof. We prove that for any arbitrary $(\mu + 1)$-world w and any formula $\varphi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m^+$: $\varphi^w \geq_p \varphi^{R(w)}$. For the same reasons as before, we only prove that $\varphi^w = \mathbf{t}$ follows from $\varphi^{R(w)} = \mathbf{t}$. We prove this by induction on the structure of φ.

- If $\varphi = P(\bar{t})$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi = \neg\psi$, $\varphi = \forall x\psi$, $\ldots$, $K_A\psi$, $E_G\psi$, $C\psi$, $O_A^\alpha\psi$ the proof is the same as for Proposition 6.1.18.
- If $\varphi = [\phi]\psi$ then
  - $\phi^w \neq \mathbf{u}$ by induction,
  - If $\phi^w = \mathbf{f}$, then $\varphi^w = \mathbf{t}$. So assume $\phi^w = \mathbf{t}$, then $\mathit{Confine}(w, \{w' | w' \in \mathcal{W}_{\mu+1} \text{ and } w' \models \phi\})$ is not empty, so take $w'' \in \mathit{Confine}(w, \{w' | w' \in \mathcal{W}_{\mu+1} \text{ and } w' \models \phi\})$. We now prove that $\varphi^w = \mathbf{t}$ by proving that $w'' \models \psi$. Since by Lemma 6.4.8:

$$\begin{aligned} &R(\mathit{Confine}(w, \{w' | w' \in \mathcal{W}_{\mu+1} \text{ and } w' \models \phi\})) \\ &\quad = \mathit{Confine}(R(w), \{R(w') | w' \in \mathcal{W}_{\mu+1} \text{ and } w' \models \phi\}) \\ &\quad = \mathit{Confine}(R(w), \{w' | w' \in \mathcal{W}_{\mu} \text{ and } w' \models \phi\}) \end{aligned}$$

    it follows from the induction hypothesis that $\varphi^w = \mathbf{t}$.

This proves that $\varphi^w = \mathbf{t}$ for all $\varphi \in \mathcal{CO}^{\omega^2}\mathcal{L}_m^+$. $\square$

Proposition 6.4.12. *For any μ-world w and μ'-world w' such that $w' \geq_p w$: for any formula formula φ in $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$: $\varphi^{w'} \geq_p \varphi^w$*

Proof. This proof is similar to the proof for $\mathcal{CO}^\gamma\mathcal{L}_m$. $\square$

Theorem 6.4.13. *A μ-world w (μ an arbitrary ordinal) can resolve every formula $\varphi \in \mathcal{CO}^\gamma\mathcal{L}_m$, with $MD(\varphi) \leq \mu$, i.e., φ^w is 2-valued for every μ-world w.*

Proof. We prove that a μ-world resolves all $\mathcal{CO}^\gamma\mathcal{L}_m$ formulas with modal depth $\leq \mu$.

First, assume that μ is zero, or is a successor ordinal. Take a μ-world w, and a formula $\varphi \in \mathcal{CO}^\gamma\mathcal{L}_m$, with $MD(\varphi) \leq \mu$. We prove that $\varphi^w \neq \mathbf{u}$ by induction on the structure of φ.

- If $\varphi = P(\bar{t})$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi = \neg\psi$, $\varphi = \forall x\psi$, $\ldots$, $K_A\psi$, $E_G\psi$, $C\psi$,$\varphi = O^\alpha_A\psi$ the proof is the same as for Proposition 6.3.8.
- If $\varphi = [\phi]\psi$, to prove that $\varphi^w \neq \mathbf{u}$, we need to asses two cases:
  - $\varphi^w = \mathbf{u}$ if $\psi^w = \mathbf{u}$, but that is not possible due to the induction hypothesis.
  - $\varphi^w = \mathbf{u}$ if $\psi^{w[\phi]} = \mathbf{u}$, but that is also not possible due to the induction hypothesis.

The proof when μ is a limit ordinal is the same as in the proof of Theorem 6.3.8. □

We define for every ordinal γ and every ordinal μ the logic μ-$\mathcal{CO}^{\omega^2}\mathcal{L}^+_m$ as the logic with language $\mathcal{CO}^{\omega^2}\mathcal{L}^+_m$ and the valuation as defined above, using μ-worlds.

Theorem 6.4.14. *Given an ordinal γ, μ-$\mathcal{CO}^\gamma\mathcal{L}_m$ has a (not necessarily strict) larger set of satisfiable formulas than μ'-$\mathcal{CO}^{\omega^2}\mathcal{L}^+_m$ if $\mu > \mu'$.*

Proof. Similar to the proof of Theorem 6.1.26. □

Consequence 6.4.1. *For any $2\omega^2$-world w the valuation $(\cdot)^w$ is two valued in $2\omega^2$-$\mathcal{CO}^{\omega^2}\mathcal{L}^+_m$.*

Proof. Direct generalisation of Consequence 6.3.3. □

Consequence 6.4.2. *There is no α-$\mathcal{CO}^{\omega^2}\mathcal{L}^+_m$ logic with a strict larger set of satisfiable formulas than $2\omega^2$-$\mathcal{CO}^{\omega^2}\mathcal{L}^+_m$.*

Proof. Direct generalisation of Consequence 6.1.3. □

6.4.3 Application: the muddy children puzzle

The muddy children puzzle brings out subtle changes to knowledge states of a group of agents. Belle and Lakemeyer used this puzzle to illustrate their logic. Since this work is an addition/improvement to the work in [Belle and Lakemeyer, 2015], we will do the same. The symbols, problem statement and nomenclature is also based on their chapter about this puzzle.

The puzzle is: Imagine n children playing together, after which they come home and some of them have mud on their face. Each child can see all the faces of the other children, but not its own. When the father sees the children, he says: "at least one of you has mud on its forehead", thus expressing a fact already known to the children. After this, he keeps asking "If you know you have mud on your forehead, please take a step forward". If k children have mud on their forehead, after k questions, all k children step forward.

To illustrate the logic $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$, we model the muddy children puzzle with n children, where all children have mud on their forehead. In this case:

$$\Sigma = \{m_1, m_2, \ldots, m_n\}$$

$$\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$$

There are n agents, the n children of the puzzle and the vocabulary contains n elements, namely one m_i for every agent A_i, saying that child i is muddy. The theory is the union of the knowledge of all agents, where each agent's knowledge can be modelled by a theory T_i:

$$\begin{aligned}
T_i = \{O_i(\bigwedge_{i\neq j} m_j \qquad & \text{\\Each agent only knows that all others are muddy}\\
\bigwedge_{i\neq j} K_j m_i \vee \bigwedge_{i\neq j} K_j \neg m_i \qquad & \text{\\All others know i'm muddy or not muddy.}\\
C(\bigvee m_j) \qquad & \text{\\It is common knowledge that}\\
& \quad \text{at least one of us is muddy}\\
)\}&
\end{aligned}$$

and $T = \bigcup_{i\in\{1..n\}} T_i$. We will shorten the unanimous denial of knowing that they have mud on their forehead of all children as:

$$No = \bigwedge_{i\in\{1..n\}} \neg K_i m_i$$

The puzzle is now to prove that after n unanimous denials of knowing that they have mud on their forehead, they all know that they have mud on their forehead.

For readability we will only prove this statement for $n = 2$, but the proof can be generalized to n larger than 2.

Theorem 6.4.15. *For every world $w \in \mathcal{W}^{pi,in}_{2\omega^2}$ such that $w \vDash \bigwedge_{i\in\{1,2\}} m_i$ and $w \vDash T_i$ for every $i \in 1, 2$ it is true that*

$$w \vDash [No] \bigwedge_{i\in\{1,2\}} K_i m_i$$

Proof. To prove this statement, take w such that the above hypothesis are satisfied. In this case, w satisfies

$$w \vDash m_1 \wedge m_2 \tag{6.1}$$

$$w \vDash O_1(m_2 \wedge (K_2 m_1 \vee K_2 \neg m_1) \wedge C(m_2 \vee m_1)) \tag{6.2}$$

$$w \vDash O_2(m_1 \wedge (K_1 m_2 \vee K_1 \neg m_2) \wedge C(m_1 \vee m_2)) \tag{6.3}$$

We will only prove that agent 1 knows he has a muddy forehead ($w \vDash [No]K_1 m_1$), the proof for agent 2 is analogous. From (6.1) we derive that $w^{obj} = \Sigma = \{m_1, m_2\}$. From (6.2) we derive that:

$$\{(w')^{obj} | w' \in A^w\} = \{\{m_1, m_2\}, \{m_2\}\}$$

since m_2 is the only objective fact that agent 1 knows.

To prove that $w \vDash [No]K_1 m_1$ it is enough to prove that $\forall v \in A^w_1$ such that $v \vDash \neg K_1 m_1 \wedge \neg K_2 m_2$ it is true that $v \vDash m_1$. Indeed, since

$$\{\{v | v \in A^w_1 \text{ and } v \vDash \neg K_1 m_1 \wedge \neg K_2 m_2\}\}$$

$$= \mathit{Confine}(w, \{w' | w' \vDash \neg K_1 m_1 \wedge \neg K_2 m_2\})$$

and since proving $w \vDash [No]K_1 m_1$ is equivalent with proving

$$\mathit{Confine}(w, \{w' | w' \vDash No\}) \vDash K_1 m_1$$

we will only prove the above statement.

Take a $v \in A^w_1$ arbitrarily such that $v \vDash \neg K_1 m_1 \wedge \neg K_2 m_2$. From (6.2) it follows that $K_2(m_1 \vee m_2)$ (since $m_1 \vee m_2$ is common knowledge). With Theorem 6.4.3 (the agents are rational), we find that $v \vDash \neg K_2 \neg m_1$. Indeed, because assume

$v \vDash K_2 \neg m_1$, then we can derive that $v \vDash K_2 m_2$ from $K_2(m_1 \vee m_2)$ and Theorem 6.4.3, which cannot be true. Using theorem 6.4.3 with this knowledge and the fact that $w' \vDash K_2 m_1 \vee K_2 \neg m_1$ (from (6.2)), we conclude that $w' \vDash K_2 m_1$ or equivalently, for every $w \in A_2^{w'} : w \vDash m_1$. Since w' is a informed world, it follows that $w' \in A_2^{w'}$ and as such that $w' \vDash m1$. This proves what was to be proven. □

6.5 Conclusion

A standard approach in modal logics is to use Kripke structures. When using Kripke structures, the set of potential worlds is fixed in advance and the logic is valuated relative to this class of worlds, by defining an access relation for each agent between the worlds. An agent knows something in a world w if it is true in all worlds accessible from w. We have shown why fixing of worlds in advance is a problem when we study only knowing. When evaluating only knowing, we want to talk about all worlds satisfying a certain formula, so we cannot afford to only talk about all worlds that are in the set of chosen worlds that satisfy that formula.

Belle and Lakemeyer defined $\mathcal{COL}_m$ in [Belle and Lakemeyer, 2015]: the first logic that studied the interaction between common knowledge and only knowing. In this chapter we analyzed their approach and proposed a new semantics, based on our findings. We proposed an improvement to the semantics, since we found anomalies in the sense of sentences that can be written in $\mathcal{COL}_m$ that should be satisfiable, but are not in their approach. It is for example not possible to define a epistemic state as defined in [Belle and Lakemeyer, 2015], where an agent only knows that something is not common knowledge ($O_A \neg Cp$).

In this chapter we first proposed a subset of that logic, $\mathcal{CL}_m$, accommodating a K and C (common knowledge) operator and used this to introduce our new proposal for a semantical structure. Key to this new proposal is to define finite worlds as tuples, infinitely deep worlds as sequences of worlds, and deeper worlds again as tuples, where agents can have such sequences as their possible worlds. We defined important monotonicity properties for these sequences to be well-defined and proved them.

We defined worlds that are deeper than ω: for any arbitrary ordinal μ we defined worlds with depth μ. By investigating two different approaches: one using Kripke structures and the approach of Belle and Lakemeyer, we show why these deeper worlds are necessary.

In our μ-worlds, some monotonicity properties we have for $\mathcal{CL}_m$ are not maintained for $\mathcal{COL}_m$. The reason for this is the special nature of the only knowing operator. To study this issue in more detail, we proposed limited only knowing. Limited only knowing states explicitly what kind of knowledge is only (or not) known and does have these monotonicity properties. We use this notion of limited only knowing to define a logic $\mathcal{CO}^{\omega^2}\mathcal{L}_m$, that does have an "unlimited" only knowing operator, but does not allow for nesting of this operator.

Other extensions to $\mathcal{COL}_m$ are defined afterwards: (positive or negative) introspective $\mathcal{CO}^{\omega^2}\mathcal{L}_m$, informed $\mathcal{CO}^{\omega^2}\mathcal{L}_m$ and $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$: $\mathcal{CO}^{\omega^2}\mathcal{L}_m$ extending with the notion of public announcements. The new logic $\mathcal{CO}^{\omega^2}\mathcal{L}_m^+$ is illustrated using the muddy children puzzle (as in [Belle and Lakemeyer, 2015]), a subtle puzzle pertaining the difference between common knowledge and everybody knowing something, using only knowing.

Interesting future work would definitely be to further explore $\mathcal{COL}_m$, $\mathcal{CO}^{\gamma}\mathcal{L}_m$ and $\mathcal{CO}^{\omega^2}\mathcal{L}_m$, to find a logic containing the O operator (without fixing a depth μ), that does not exhibit the issues in previous approaches, has the monotonicity properties and can be nested without problems. Further, complexity results have to be investigated as well as interesting inferences or a decision procedure to use (fragments of) this logic for real-life applications.

7 Conclusion

The goal of the research presented in this thesis was to investigate the feasability of using the KBS IDP to solve problems in different situations, ranging from practical situations as executing database related tasks on a database containing cars and planning reservations for rentals of those cars, and supporting an interactive configuration system, to more theoretical situations where a group of agents together decide who gets access to a resource by referring to each others knowledge. As discussed in Chapter 1, a good KBS has a rich language that can express a wide range of knowledge in a natural way and supports a large set of reasoning tasks. On both fronts we did research to investigate the current state of the IDP KBS and proposed extensions to the knowledge representation language FO(·) to be able to express the relevant domain knowledge and new inferences to support all needed reasoning tasks.

We studied multiple applications from different contexts such as interactive configuration, Business Rules and Access Control and proposed new language constructs and/or reasoning tasks for each of them.

- In Chapter 3 we studied interactive configuration and discussed why solutions for interactive configuration problems are hard to develop using standard (declarative) paradigms. We identified 8 different subtasks relevant to interactive configuration and formalized these as logical inferences on a knowledge base. Using this, a system was built with a centrally maintained knowledge base containing all relevant configuration knowledge of the domain at hand and with support for the reasoning tasks. We tested this as a proof of concept for a banking company. To evaluate, ten other existing approaches were discussed and compared to our proposal using a set of criteria used in knowledge-based configuration literature. The proof of concept was very promising and opened perspectives for future research. There are multiple language extensions, interesting for representing configuration knowledge that can be formalized as extensions of the FO($\cdot$) knowledge representation language. Some examples are templates [Dasseville et al., 2015] that allow to create extra abstraction layers, and reification which adds a meta level to reason about the symbol instead of its interpretation (to for example assign a symbol to a class like “Administrative” or “Technical”).

- Another domain where we studied the feasability of the knowledge base paradigm was the domain of Business Rules. We studied 2 usecases of the EU-Rent Car Rental company: scheduling a set of reservations and processing small database changes such as adding a new car to the domain. Advantages of a logic based approach in the IDP KBS were readability, maintainability and reusability, and for this application we found another specific advantage: the inherent non-determinism of a logic based approach allowed to select a more optimal solution for scheduling then the deterministic rule-based solution. We proposed one new derived inference and formalised a language extension: FO($\cdot^+$), which extended the notion of definitions and allows disjunctions and the “new”-operator in the head, such that knowledge can be specified about situations where a new domain element is created. This work was a first step of research, and since then a lot of further work has been done. The new language FO($\cdot^+$) has been further extended and explored and resulted in FO(C) [Bogaerts et al., 2014c]. An interesting further research direction is developing logical inferences to use with FO(C) theories and to implement those so that they can be used in a KBS.

- Motivated by an application of access control in a distributed setting we developed *dAEL: distributed autoepistemic logic*, a generalisation of autoepistemic logic (AEL). The syntax was formulated by replacing the K operator in AEL by a set of K_A operators, one for each agent A. Assuming the theories of the different agents to be characterizations of there public commitments, we made dAEL a logic with full (positive and negative) mutual introspection. This means that every agent can refer to the knowledge (or equivalently in our framework: the consequences of the commitments) of other agents and the lack thereof. Using approximation fixpoint theory, we generalized 5 different semantics from the single agent setting to dAEL. One possible future research direction is generalising more semantics that are studied in the context of AEL to the distributed case. Other interesting future work (which we are currently doing in the context of a journal publication on this subject) is studying a decision procedure for dAEL(ID), so that it can be used for practical applications.

- $\mathcal{COL}_m$ is a multi-agent modal logic, that does not assume introspection between agents and has operators for knowledge, only knowing and common knowledge. It is a more general modal logic then dAEL, but is also has a more complex semantics. In Chapter 6, we propose a new semantical structure for $\mathcal{COL}_m$. The motivation behind proposing these new semantics are anomalies we found in earlier approaches. We show these anomalies and explain why they originate and show ideas how to solve them. We also extend the language $\mathcal{COL}_m$ to a language $\mathcal{COL}_m^+$ that can model public announcements and illustrate that logic using the well-known muddy children puzzle. While we have a set of worlds that can interpret every formula in $\mathcal{CL}_m$ (the set of all formulas in $\mathcal{COL}_m$ not containing only knowing) and given a formula in $\mathcal{COL}_m$, we have a set of world that can interpret that formula: we do not have a set of worlds that can interpret every formula in $\mathcal{COL}_m$. This is important future work, to be done in the near future. Another interesting direction of future work is the study of other semantics, especially only knowing. Using approximation fixpoint theory, we can study generalisations of for example the well-founded and the stable semantics for $\mathcal{COL}_m$. For this language, no implementation of inferences has been made, so it might be interesting to study the complexity of inferences for $\mathcal{COL}_m$ and implement some inferences for (a subset of) $\mathcal{COL}_m$.

To conclude, I feel that the knowledge base paradigm is a strong approach and the IDP system is becoming a mature implementation of it. The knowledge base paradigm has many advantages over other software development paradigms (both imperative and declarative) and there are a lot of applications where these advantages are really clear and important, as we have shown in this thesis. The time is near to start solving real-life applications, not only as a prototype or proof of concept, but to build real applications based around a central knowledge base. Especially in the context of applications such as interactive configuration we studied in this thesis, the knowledge base paradigm can really contribute and is mature enough for real applications, as we have shown. For the multi-agent applications in this thesis, the research and work done was more theoretical, and while these approaches are also promising, there is more research needed before these languages are ready for real applications. We studied them from a theoretical point of view, and our main goal was to develop semantics for these new logics. However, when inferences for these logics are implemented, they will fit perfectly in a KBS and will open a whole new field of applications for the IDP KBS to be applied in.

A Appendix

A.1 Proofs for Section 5.2

In order to prove Theorems 5.2.12 and 5.2.16 we first need to define some additional notions and prove some lemmas.

Recall that we use the following notational conventions: ϕ denotes a dAEL formula over Σ , φ denotes an AEL formula over Σ', I denotes a Σ-structure, and J denotes a Σ'-structure. We defined every τ_x ($x \in \{\varphi, T, J, Q, B\}$) as a translation function, with a subscript x identifying what it translates to. For example: τ_J translates a indexed family $\mathcal{I} = (I_A)_{A \in \mathcal{A}}$ of structures to a Σ' structure J.

Definition A.1.1. Given a Σ'-structure (in dAEL) J and an agent A, we write J_A for the (AEL) Σ-structure defined by $s^{J_A}(d_1, \dots, d_n) := s^J(d_1, \dots, d_n, A) = (s_A)^J$ for every $s \in \Sigma$.

A Σ' structure J represents a state of affairs of the world, from the view of all agents. J_A is a projection of this representation to the state of the world from the view of agent A.

Next lemma states that the this projection is an inverse to the translation of distributed possible world structures: the conservative view of agent A in a dbp is equal to the projection to agent A of the the translation of all conservative views.

Lemma A.1.2. *When a distributed belief pair $\mathcal{B}$ is universally consistent:*

$$\{J_{A'} | J \in \tau_Q(\mathcal{B}^c)\} = \mathcal{B}^c_{A'}$$

for each $A' \in \mathcal{A}$.

Proof. We prove the equality by proving the subset relation in both directions.

Let $J' \in \tau_Q(\mathcal{B}^c)$. Then there is an indexed family $(I_A)_{A\in\mathcal{A}}$ s.t. $I_A \in \mathcal{B}^c_A$ for each $A \in \mathcal{A}$ and $J' = \tau_J((I_A)_{A\in\mathcal{A}})$. Then for each interpretation $f^{J'_{A'}}(d_1, \dots, d_n) = f^{J'}(d_1, \dots, d_n, A') = f^{I_{A'}}(d_1, \dots, d_n)$. So $J'_{A'} = I_{A'}$ and $I_{A'} \in \mathcal{B}^c_{A'}$, as required.

To prove the other direction, let $I \in \mathcal{B}^c_{A'}$. Since $\mathcal{B}^c$ is universally consistent there is some indexed family $(I_A)_{A\in\mathcal{A}}$ s. t. $I_A \in \mathcal{B}^c_A$ for all $A \in \mathcal{A}$ and $I_{A'} = I$. Define $J' := \tau_J((I_A)_{A\in\mathcal{A}}$. Then $J'_{A'} = I_{A'} = I$, as required. □

The following lemma says that the mapping is faithful to the valuations of AEL and dAEL formulas:

Lemma A.1.3. *For an agent A, a formula $\phi \in \mathcal{L}_d$, a* universally consistent *distributed belief pair $\mathcal{B}$ and a Σ'-structure J,*

$$\tau_\varphi(A, \phi)^{\tau_B(\mathcal{B}),J} = \phi^{\mathcal{B},J_A}.$$

Proof. We prove the lemma by induction over the structure of ϕ.

- Assume $\phi = P(t_1, \dots, t_n)$. Then $\tau_\varphi(A, \phi)^{\tau_B(\mathcal{B}),J} = \mathbf{t}$ if $(t_{1A}, \dots, t_{nA}, A) \in P^J$ which is by definition of the translation of structures the same as $(t_1^A, \dots, t_n^A, A) \in P^{J_A}$. Analogously, we find $\tau_\varphi(t, \phi)^{\tau_B(\mathcal{B}),J} = \mathbf{f}$ iff $\phi^{\mathcal{B},J_A} = \mathbf{f}$.
- Assume $\phi = \neg\psi$. Then $\tau_\varphi(t, \phi)^{\tau_B(\mathcal{B}),J} = \neg\tau_\varphi(t, \psi)^{\tau_B(\mathcal{B}),J}$ and the result follows by the induction hypothesis.
- If $\phi = \phi_1 \wedge \phi_2$ or $\phi = (\forall x)\psi$, the proof is similar.

- Assume $\phi = K_{A'}\psi$, then $\tau_\varphi(A,\phi)^{\tau_B(\mathcal{B}),J} = \mathbf{t}$ if and only if $(K\tau_\varphi(A',\psi))^{\tau_B(\mathcal{B}),J} = \mathbf{t}$, and

$$\begin{aligned}\tau_\varphi(A,\phi)^{\tau_B(\mathcal{B}),J} = \mathbf{t} &\Leftrightarrow (K\tau_\varphi(A',\psi))^{\tau_B(\mathcal{B}),J} = \mathbf{t}\\ &\Leftrightarrow \text{for each } J' \in \tau_Q(\mathcal{B}^c)\text{: } \tau_\varphi(A',\psi)^{\tau_B(\mathcal{B}),J'} = \mathbf{t}\\ &\qquad \text{(Definition 5.2.4)}\\ &\Leftrightarrow \text{for each } J' \in \tau_Q(\mathcal{B}^c)\text{: } \psi^{\mathcal{B},J'_{A'}} = \mathbf{t}\\ &\qquad \text{(by the induction hypothesis)}\\ &\Leftrightarrow \text{for each } I \in \mathcal{B}^c_{A'}\text{: } \psi^{\mathcal{B},I} = \mathbf{t}\\ &\qquad \text{(by Lemma A.1.2)}\\ &\Leftrightarrow \phi^{\mathcal{B},J_A} = \mathbf{t}\end{aligned}$$

□

The following lemma states that the dAEL approximator $\mathcal{D}^*_\mathcal{T}$ is mapped to the AEL approximator $D^*_{\tau_T(\mathcal{T})}$, when restricted to universally consistent distributed belief pairs:

Lemma A.1.4. *For every distributed theory $\mathcal{T}$ and every universally consistent distributed belief pair $\mathcal{B}$,*

$$\tau_B(\mathcal{D}^*_\mathcal{T}(\mathcal{B})) = D^*_{\tau_T(\mathcal{T})}(\tau_B(\mathcal{B})).$$

Proof. $\tau_B(\mathcal{D}^*_{\mathcal{T}}(\mathcal{B}))$

$$\begin{aligned}
&= (\tau_Q((\{I \mid \phi^{\mathcal{B},I} = \mathbf{t} \text{ for each } \phi \in T_A\})_{A\in\mathcal{A}}),\\
&\quad \tau_Q((\{I \mid \phi^{\mathcal{B},I} \neq \mathbf{f} \text{ for each } \phi \in T_A\})_{A\in\mathcal{A}}))\\
&= (\{J \mid \phi^{\mathcal{B},J_A} = \mathbf{t} \text{ for each } A \in \mathcal{A} \text{ and } \phi \in T_A\},\\
&\quad \{J \mid \phi^{\mathcal{B},J_A} \neq \mathbf{f} \text{ for each } A \in \mathcal{A} \text{ and } \phi \in T_A\})\\
&= (\{J \mid \tau_\varphi(A,\phi)^{\tau_B(\mathcal{B}),J} = \mathbf{t} \text{ for each } A \in \mathcal{A}\\
&\qquad \text{and } \phi \in T_A\},\\
&\quad \{J \mid \tau_\varphi(A,\phi)^{\tau_B(\mathcal{B}),J} \neq \mathbf{f} \text{ for each } A \in \mathcal{A}\\
&\qquad \text{and } \phi \in T_A\}) \text{ (by Lemma A.1.3)}\\
&= (\{J \mid \varphi^{\tau_B(\mathcal{B}),J} = \mathbf{t} \text{ for each } \varphi \in \tau_T(\mathcal{T})\},\\
&\quad \{J \mid \varphi^{\tau_B(\mathcal{B}),J} \neq \mathbf{f} \text{ for each } \varphi \in \tau_T(\mathcal{T})\})\\
&= \mathcal{D}^*_{\tau_T(\mathcal{T})}(\tau_B(\mathcal{B}))
\end{aligned}$$
□

The following lemma is a restriction of Lemma A.1.4 to the conservative operators $\mathcal{D}^c_{\mathcal{T}}$ and $\mathcal{D}^l_{\mathcal{T}}$:

Lemma A.1.5. *For every distributed theory $\mathcal{T}$ and all universally consistent DPWS's $\mathcal{Q}$ and $\mathcal{Q}'$,*

$$\tau_Q(\mathcal{D}^c_{\mathcal{T}}(\mathcal{Q}', \mathcal{Q})) = \mathcal{D}^c_{\tau_T(\mathcal{T})}(\tau_Q(\mathcal{Q}'), \tau_Q(\mathcal{Q})).$$

Proof. Trivial by Lemma A.1.4. □

The mapping maps the dAEL knowledge revision operator $\mathcal{D}_{\mathcal{T}}$ to the corresponding AEL knowledge revision operator $\mathcal{D}_T$:

Lemma A.1.6. *For every distributed theory $\mathcal{T}$ and every DPWS $\mathcal{Q}$,*

$$\tau_Q(\mathcal{D}_{\mathcal{T}}(\mathcal{Q})) = \mathcal{D}_{\tau_T(\mathcal{T})}(\tau_Q(\mathcal{Q})).$$

Proof. Follows from Lemma A.1.4 and the fact that $D^*_T(\mathcal{Q}, \mathcal{Q}) = (D_T(\mathcal{Q}), D_T(\mathcal{Q}))$ for each $\mathcal{Q}$. □

The mapping is faithful to the (universal) consistency of (distributed) possible world structures:

Lemma A.1.7. *A DPWS $\mathcal{Q}$ is universally consistent iff $(\tau_Q(\mathcal{Q}))_A \neq \emptyset$ for any $A \in \mathcal{A}$.*

Proof. Trivial. □

The following lemma states that the restriction of τ_Q to universally consistent DPWS's is injective:

Lemma A.1.8. *If $\mathcal{Q}$ and $\mathcal{Q}'$ are DPWS's such that $\mathcal{Q}$ is universally consistent and $\tau_Q(\mathcal{Q}) = \tau_Q(\mathcal{Q}')$, then $\mathcal{Q} = \mathcal{Q}'$.*

Proof. By Lemma A.1.7, $\mathcal{Q}'$ is universally consistent too. By symmetry, it is enough to show that $\mathcal{Q}_A \subseteq \mathcal{Q}'_A$ for all $A \in \mathcal{A}$.

So let $I_{A'} \in \mathcal{Q}_{A'}$. Given that $\mathcal{Q}$ is universally consistent, we can choose an $I_B \in \mathcal{Q}_B$ for every $B \in \mathcal{A} \setminus \{A'\}$. Then $\tau_J((I_A)_{A \in \mathcal{A}}) \in \tau_Q(\mathcal{Q}) = \tau_Q(\mathcal{Q}')$, so $I_A \in \mathcal{Q}'_A$, as required. □

The same holds for τ_B on universally consistent distributed belief pairs:

Lemma A.1.9. *If $\mathcal{B}$ and $\mathcal{B}'$ are DPWS's such that $\mathcal{B}$ is universally consistent and $\tau_Q(\mathcal{B}) = \tau_Q(\mathcal{B}')$, then $\mathcal{B} = \mathcal{B}'$.*

Proof. Similar to proof of Lemma A.1.8. □

The mapping is faithful to the knowledge order:

Lemma A.1.10. *If $\mathcal{Q} \leq_K \mathcal{Q}'$, then $\tau_Q(\mathcal{Q}) \leq_K \tau_Q(\mathcal{Q}')$.*

Proof. Let $J \in \tau_Q(\mathcal{Q}')$, i.e., for every $A \in \mathcal{A}$, $J_A \in \mathcal{Q}'_A$, as such $J_A \in \mathcal{Q}_A$. So $J \in \tau_Q(\mathcal{Q})$. □

The following lemma states that the mapping is faithful to $\leq_K$-least upper bounds and greatest lower bounds:

Lemma A.1.11. *For a set $\mathcal{S}$ of DPWS's, $\tau_Q(\mathrm{lub}_{\leq_K}(\mathcal{S})) = lub_{\leq_K}(\tau_Q[\mathcal{S}])$ and $\tau_Q(\mathrm{glb}_{\leq_K}(\mathcal{S})) = \mathrm{glb}_{\leq_K}(\tau_Q[\mathcal{S}])$.*

Proof. We prove the first equality; the second one can be proven similarly.

First we show that $\tau_Q(\text{lub}(\mathcal{S}))$ is an upper bound of $\tau_Q[\mathcal{S}]$: Let $Q \in \tau_Q[\mathcal{S}]$. Then there is a DPWS $\mathcal{Q} \in \mathcal{S}$ such that $Q = \tau_Q(\mathcal{Q})$. Since $\mathcal{Q} \leq_K \text{lub}\,\mathcal{S}$, Lemma A.1.10 implies that $Q \leq_K \tau_Q(\text{lub}(\mathcal{S}))$.

Now we show that for each upper bound Q' of $\tau_Q[\mathcal{S}]$, $\tau_Q(\text{lub}(\mathcal{S})) \leq_K Q'$: Suppose that for every $Q \in \tau_Q[\mathcal{S}]$, $Q \leq_K Q'$, i.e.,$Q' \subseteq Q$. We need to show that $\tau_Q(\text{lub}(\mathcal{S})) \leq_K Q'$, i.e.,that $Q' \subseteq \tau_Q(\text{lub}(\mathcal{S}))$. So let $J \in Q'$. Let $\mathcal{Q} \in \mathcal{S}$. Then $\tau_Q(\mathcal{Q}) \in \tau_Q[\mathcal{S}]$, so $Q' \subseteq \tau_Q(\mathcal{Q})$. Hence $J \in \tau_Q(\mathcal{Q})$, i.e.,$J \mid_A \in \mathcal{Q}_A$ for each $A \in \mathcal{A}$. Given that $\mathcal{Q}$ was an arbitrary element of $\mathcal{S}$, we have that $J \mid_A \in \bigcap\{Q \mid \text{for some } \mathcal{Q} \in \mathcal{S}, Q = \mathcal{Q}_A\}$. So $J \in \tau_Q((\bigcap\{Q \mid \text{for some } \mathcal{Q} \in \mathcal{S}, Q = \mathcal{Q}_A\})_{A\in\mathcal{A}}) = \tau_Q(\text{lub}(\mathcal{S}))$, as required. □

The mapping is faithful to $\leq_p$-least upper bounds:

Lemma A.1.12. *For a set $\mathcal{S}$ of distributed belief pairs, $\tau_B(\text{lub}_{\leq_p}(\mathcal{S})) = \text{lub}_{\leq_p}(\tau_B[\mathcal{S}])$.*

Proof. This follows directly from Lemma A.1.10 and Lemma A.1.11. □

The following states that the stable revision of an element of the image of τ_Q is itself in the image of τ_Q:

Lemma A.1.13. *For any DPWS $\mathcal{Q}$, there is a DPWS $\mathcal{Q}'$ such that $\tau_Q(\mathcal{Q}') = \mathcal{D}^{st}_{\tau_T(T)}(\tau_Q(\mathcal{Q}))$.*

Proof. Let α be such that $\mathcal{D}^{st}_{\tau_T(T)}(\tau_Q(\mathcal{Q})) = \mathcal{D}^{c}_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))^{\alpha}(\bot)$. By induction, it is enough to show that for each ordinal number α, there is a DPWS $\mathcal{Q}'$ such that $\tau_Q(\mathcal{Q}') = \mathcal{D}^{c}_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))^{\alpha}(\bot)$.

For $\alpha = 0$, let $\mathcal{Q}' := (\bot)_{A\in\mathcal{A}}$. Then $\tau_Q(\mathcal{Q}') = (\bot) = \mathcal{D}^{c}_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))^{0}(\bot)$.

Suppose the result holds for α, i.e. there is a DPWS $\mathcal{Q}'$ such that $\tau_Q(\mathcal{Q}') = \mathcal{D}^{c}_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))^{\alpha}(\bot)$. By Lemma A.1.5, $\tau_Q(\mathcal{D}^{c}_{T}(\mathcal{Q}', \mathcal{Q})) = \mathcal{D}^{c}_{\tau_T(T)}(\tau_Q(\mathcal{Q}'), \tau_Q(\mathcal{Q})) = \mathcal{D}^{c}_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))^{\alpha+1}(\bot)$.

Let λ be a limit ordinal such that the result holds for every $\alpha < \lambda$. Define $\mathcal{S} := \{\mathcal{Q}' \mid \tau_Q(\mathcal{Q}') = \mathcal{D}^{c}_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))^{\alpha}(\bot) \text{ for some } \alpha < \lambda\}$. By Lemma A.1.11, $\tau_Q(\text{lub}(\mathcal{S})) = \text{lub}(\tau_Q[\mathcal{S}]) = \mathcal{D}^{c}_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))^{\lambda}(\bot)$. □

The following lemma states that the mapping maps the stable dAEL knowledge revision operator $\mathcal{D}^{st}_{T}$ to the stable AEL knowledge revision operator $\mathcal{D}^{st}_{T}$:

Lemma A.1.14. *For every universally consistent distributed theory $\mathcal{T}$ and every DPWS $\mathcal{Q}$,*

$$\mathcal{D}^{st}_{\tau_T(T)}(\tau_Q(\mathcal{Q})) = \tau_Q(\mathcal{D}^{st}_T(\mathcal{Q})).$$

Proof. Let Q denote $\mathcal{D}^{st}_{\tau_T(T)}(\tau_Q(\mathcal{Q}))$.

First suppose $Q = \emptyset$. We need to show that $\tau_Q(\mathcal{D}^{st}_T(\mathcal{Q})) = \emptyset$, i.e.,that $\mathcal{D}^{st}_T(\mathcal{Q}) = \text{lfp}(\mathcal{D}^c_T(\cdot, \mathcal{Q}))$ is not universally consistent. For this it is enough to show that every universally consistent DPWS is not a fixpoint of $\mathcal{D}^c_T(\cdot, \mathcal{Q})$. So suppose $\mathcal{Q}'$ is universally consistent. Then $\tau_Q(\mathcal{Q}') \neq \emptyset$, i.e.,$\tau_Q(\mathcal{Q}') <_K Q$. Since Q is the least fixpoint of $\mathcal{D}^c_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))$, $\mathcal{D}^c_{\tau_T(T)}(\tau_Q(\mathcal{Q}'), \tau_Q(\mathcal{Q})) \neq \tau_Q(\mathcal{Q}')$. So by Lemma A.1.5, $\tau_Q(\mathcal{D}^c_T(\mathcal{Q}', \mathcal{Q})) \neq \tau_Q(\mathcal{Q}')$, i.e. $\mathcal{D}^c_T(\mathcal{Q}', \mathcal{Q}) \neq \mathcal{Q}'$, as required.

Now suppose $Q \neq \emptyset$. By Lemma A.1.13, there is a DPWS $\mathcal{Q}'$ such that $\tau_Q(\mathcal{Q}') = Q$. Note that by Lemma A.1.7, $\mathcal{Q}$ is universally consistent. $Q = \tau_Q(\mathcal{Q}')$ is a fixpoint of $\mathcal{D}^c_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))$, i.e. $\mathcal{D}^c_{\tau_T(T)}(\tau_Q(\mathcal{Q}'), \tau_Q(\mathcal{Q})) = \tau_Q(\mathcal{Q}')$. By Lemma A.1.5, $\tau_Q(\mathcal{D}^c_T(\mathcal{Q}', \mathcal{Q})) = \tau_Q(\mathcal{Q}')$. By Lemma A.1.8, $\mathcal{D}^c_T(\mathcal{Q}', \mathcal{Q}) = \mathcal{Q}'$, i.e. $\mathcal{Q}'$ is a fixpoint of $\mathcal{D}^c_T(\cdot, \mathcal{Q})$. Let $\mathcal{Q}''$ denote the least fixpoint of $\mathcal{D}^c_T(\cdot, \mathcal{Q})$. Then $\mathcal{Q}'' \leq_K \mathcal{Q}'$, so by Lemma A.1.10, $\tau_Q(\mathcal{Q}'') \leq_K \tau_Q(\mathcal{Q}')$. Additionally, $\mathcal{D}^c_T(\mathcal{Q}'', \mathcal{Q}) = \mathcal{Q}''$, so $\tau_Q(\mathcal{D}^c_T(\mathcal{Q}'', \mathcal{Q})) = \tau_Q(\mathcal{Q}'')$, so by Lemma A.1.5, $\tau_Q(\mathcal{Q}'')$ is a fixpoint of $\mathcal{D}^c_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))$. Since $Q = \tau_Q(\mathcal{Q}')$ is the least fixpoint of $\mathcal{D}^c_{\tau_T(T)}(\cdot, \tau_Q(\mathcal{Q}))$, $\tau_Q(\mathcal{Q}') \leq_K \tau_Q(\mathcal{Q}'')$. Combining the two inequalities, we get $\tau_Q(\mathcal{Q}') = \tau_Q(\mathcal{Q}'')$, so by Lemma A.1.8, $\mathcal{Q}' = \mathcal{Q}''$. So $\mathcal{Q}' = \text{lfp}(\mathcal{D}^c_T(\cdot, \mathcal{Q})) = \mathcal{D}^{st}_T(\mathcal{Q})$, i.e. $Q = \tau_Q(\mathcal{Q}') = \tau_Q(\mathcal{D}^{st}_T(\mathcal{Q}))$, as required. □

Lemma A.1.15. *If $\mathcal{T}$ is permaconsistent, then $\mathcal{D}^*_{\mathcal{T}}(\bot, \top)$ is universally consistent.*

Proof. If $\mathcal{T}$ is permaconsistent, then for each agent $A \in \mathcal{A}$ and each theory T' that can be constructed from $T_A = (\mathcal{T})_A$ by replacing any non-nested occurrence of modal literals by $\mathbf{t}$ or $\mathbf{f}$ is consistent. Now, for each agent A, let T'_A be the theory constructed from T_A by replacing all non-nested occurrences of modal literals by $\mathbf{t}$ if they occur in a negative context (under an odd number of negations) and by $\mathbf{f}$ otherwise. This theory is clearly stronger than T_A. Since $\mathcal{T}$ is permaconsistent, T'_A is satisfiable, so let I_A be a model of T'_A. In this case, it holds that $T_A^{(\bot,\top),I_A} = \mathbf{t}$ (since T_A is weaker than T'_A).

From this, we find that for each agent A, $\{I \mid T_A^{(\bot,\top),I}\}$ is non-empty and thus that $\mathcal{D}^*_{\mathcal{T}}{}^*(\bot, \top)$ is indeed universally consistent. □

We are now ready to present the proofs of Theorems 5.2.12 and 5.2.16.

Proof of Theorem 5.2.16.

Case 1: $\sigma = \mathsf{Sup}$: Suppose $\mathcal{Q}$ is a universally consistent DPWS. $\mathcal{Q}$ is a Sup-model of T
iff $\mathcal{D}_{\mathcal{T}}(\mathcal{Q}) = \mathcal{Q}$
iff $\tau_Q(\mathcal{D}_{\mathcal{T}}(\mathcal{Q})) = \tau_Q(\mathcal{Q})$ by Lemma A.1.8
iff $\mathcal{D}_{\tau_T(T)}(\tau_Q(\mathcal{Q})) = \tau_Q(\mathcal{Q})$ by Lemma A.1.6
iff $\tau_Q(\mathcal{Q})$ is a Sup-model of $\tau_T(T)$.

Case 2: $\sigma = \mathsf{PSt}$: Similar to Case 1, but using Lemma A.1.9 instead of Lemma A.1.8 and Lemma A.1.14 instead of Lemma A.1.6.

Case 3: $\sigma = \mathsf{St}$: follows from Case 2 since St-models are two-valued PSt-models.

□

Proof of Theorem 5.2.12.

Case 1: $\sigma \in \{\mathsf{Sup}, \mathsf{PSt}, \mathsf{St}\}$: follows by combining Theorems 5.2.16 and 5.2.17.

Case 2: $\sigma = \mathsf{KK}$: The KK-model of T is the $\leq_p$-least fixpoint of $\mathcal{D}^*_T$ and the KK-model of $\tau_T(T)$ is the $\leq_p$-least fixpoint of $\mathcal{D}^*_{\tau_T(T)}$. So by Lemma A.1.8, it is enough to show that for each ordinal number $\alpha > 0$, ${\mathcal{D}^*_T}^\alpha((\bot,\top)_{A\in\mathcal{A}}$ is universally consistent and $\tau_B({\mathcal{D}^*_T}^\alpha((\bot,\top)_{A\in\mathcal{A}}) = {\mathcal{D}^*_{\tau_T(T)}}^\alpha((\bot,\top))$. We prove this by transfinite induction.

For $\alpha = 1$, this is follows from Lemma A.1.15.

Suppose it is true for α. Then

$$\begin{aligned}\tau_B({\mathcal{D}^*_T}^{\alpha+1}((\bot,\top)_{A\in\mathcal{A}}) &= \tau_B(\mathcal{D}^*_T({\mathcal{D}^*_T}^\alpha((\bot,\top)_{A\in\mathcal{A}}))\\ &= \mathcal{D}^*_{\tau_T(T)}(\tau_B({\mathcal{D}^*_T}^\alpha((\bot,\top)_{A\in\mathcal{A}})) \text{ by Lemma A.1.4}\\ &= {\mathcal{D}^*_{\tau_T(T)}}^{\alpha+1}((\bot,\top)) \text{ by assumption about } \alpha.\end{aligned}$$

Now suppose it is true for all $\alpha < \lambda$. Then

$$\begin{aligned}&\tau_B({\mathcal{D}^*_T}^\lambda((\bot,\top)_{A\in\mathcal{A}}))\\ &= \tau_B(\mathrm{lub}(\{{\mathcal{D}^*_T}^\alpha((\bot,\top)_{A\in\mathcal{A}}) \mid \alpha < \lambda\}))\\ &= \mathrm{lub}(\tau_B[\{{\mathcal{D}^*_T}^\alpha((\bot,\top)_{A\in\mathcal{A}}) \mid \alpha < \lambda\}]) \text{ by Lemma A.1.12}\\ &= \mathrm{lub}(\{{\mathcal{D}^*_{\tau_T(T)}}^\alpha((\bot,\top)) \mid \alpha < \lambda\})\\ &= {\mathcal{D}^*_{\tau_T(T)}}^\lambda((\bot,\top))\end{aligned}$$

Case 3: $\sigma = \mathsf{WF}$: Similar to Case 2, but using Lemma A.1.14 instead of Lemma A.1.4. $\square$

Bibliography

M. Abadi. Variations in Access Control Logic. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science*, pages 228–233, 2003.

M. Abadi. Variations in Access Control Logic. In *9th International Conference on Deontic Logic in Computer Science*, pages 96–109, 2008.

M. Abadi, M. Burrows, B. W. Lampson, and G. D. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15 (4):706–734, 1993. doi: 10.1145/155183.155225. URL `http://doi.acm.org/10.1145/155183.155225`.

S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *J. Comput. Syst. Sci.*, 43(1):62–124, 1991. doi: 10.1016/0022-0000(91)90032-Z. URL `http://dx.doi.org/10.1016/0022-0000(91)90032-Z`.

Adaptive Planet. Adaptive planet. `http://www.adaptiveplanet.com/`, 2015.

A. Agrawal. Semantics of business process vocabulary and process rules. In *ISEC*, pages 61–68. ACM, 2011.

G. Aucher and V. Belle. Multi-agent only knowing on planet kripke. In Yang and Wooldridge [2015], pages 2713–2719. ISBN 978-1-57735-738-4. URL `http://ijcai.org/Abstract/15/384`.

T. Axling and S. Haridi. A tool for developing interactive configuration applications. *Journal of Logic Programming*, 26(2):147–168, 1996.

doi: 10.1016/0743-1066(95)00097-6. URL http://dx.doi.org/10.1016/0743-1066(95)00097-6.

R. C. Barcan. A functional calculus of first order based on strict implication. *J. Symb. Log.*, 11(1):1–16, 1946. doi: 10.2307/2269159. URL http://dx.doi.org/10.2307/2269159.

V. E. Barker and D. E. O'Connor. Expert systems for configuration at digital: XCON and beyond. *Commun. ACM*, 32(3):298–318, 1989. doi: 10.1145/62065.62067. URL http://doi.acm.org/10.1145/62065.62067.

V. Belle and G. Lakemeyer. Multi-agent only-knowing revisited. In F. Lin, U. Sattler, and M. Truszczyński, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010.* AAAI Press, 2010. URL http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1361.

V. Belle and G. Lakemeyer. Only knowing meets common knowledge. In Yang and Wooldridge [2015], pages 2755–2761. ISBN 978-1-57735-738-4. URL http://ijcai.org/papers15/Abstracts/IJCAI15-390.html.

B. Bogaerts. *Groundedness in logics with a fixpoint semantics.* PhD thesis, Department of Computer Science, KU Leuven, June 2015. URL https://lirias.kuleuven.be/handle/123456789/496543. Denecker, Marc (supervisor), Vennekens, Joost and Van den Bussche, Jan (cosupervisors).

B. Bogaerts, J. Jansen, M. Bruynooghe, B. De Cat, J. Vennekens, and M. Denecker. Simulating dynamic systems using linear time calculus theories. *TPLP*, 14(4–5):477–492, 7 2014a. ISSN 1475-3081. doi: 10.1017/S1471068414000155. URL http://journals.cambridge.org/article_S1471068414000155.

B. Bogaerts, J. Vennekens, M. Denecker, and J. Van den Bussche. Inference in the FO(C) modelling language. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 111–116. IOS Press, 2014b. doi: 10.3233/978-1-61499-419-0-111. URL http://dx.doi.org/10.3233/978-1-61499-419-0-111.

B. Bogaerts, J. Vennekens, M. Denecker, and J. Van den Bussche. FO(C): A knowledge representation language of causality. *TPLP*, 14(4–5-Online-Supplement):60–69, 2014c. URL https://lirias.kuleuven.be/handle/123456789/459436.

M. Bruynooghe, H. Blockeel, B. Bogaerts, B. De Cat, S. De Pooter, J. Jansen, A. Labarre, J. Ramon, M. Denecker, and S. Verwer. Predicate logic as a modeling language: modeling and solving some machine learning and data mining problems with IDP3. *TPLP*, 15:783–817, November 2015. ISSN 1475-3081. doi: 10.1017/S147106841400009X. URL http://journals.cambridge.org/article_S147106841400009X.

F. Calimeri, G. Ianni, and F. Ricca. The third open answer set programming competition. *TPLP*, 14(1):117–135, 2014. doi: 10.1017/S1471068412000105. URL http://dx.doi.org/10.1017/S1471068412000105.

M. Cramer, D. A. Ambrossio, and P. Van Hertum. A logic of trust for reasoning about delegation and revocation. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies* , pages 173–184. ACM, 2015.

I. Dasseville, M. van der Hallen, G. Janssens, and M. Denecker. Semantics of templates in a compositional framework for building logics. *TPLP*, 15(4–5): 681–695, 2015. doi: 10.1017/S1471068415000319. URL http://dx.doi.org/10.1017/S1471068415000319.

B. De Cat, B. Bogaerts, M. Bruynooghe, G. Janssens, and M. Denecker. Predicate logic as a modelling language: The IDP system. *CoRR*, abs/1401.6312v2, 2016. URL http://arxiv.org/abs/1401.6312v2.

M. Denecker. The well-founded semantics is the principle of inductive definition. In J. Dix, L. F. del Cerro, and U. Furbach, editors, *JELIA*, volume 1489 of *LNCS*, pages 1–16. Springer, 1998. ISBN 3-540-65141-1.

M. Denecker. Extending classical logic with inductive definitions. In J. W. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, editors, *CL*, volume 1861 of *LNCS*, pages 703–717. Springer, 2000. ISBN 3-540-67797-6.

M. Denecker and E. Ternovska. A logic of nonmonotone inductive definitions. *ACM Trans. Comput. Log.*, 9(2):14:1–14:52, Apr. 2008. ISSN 1529-3785. URL http://dx.doi.org/10.1145/1342991.1342998.

M. Denecker and J. Vennekens. Building a knowledge base system for an integration of logic programming and classical logic. In M. García de la Banda and E. Pontelli, editors, *ICLP*, volume 5366 of *LNCS*, pages 71–76. Springer, 2008. ISBN 978-3-540-89981-5. URL http://dx.doi.org/10.1007/978-3-540-89982-2_12.

M. Denecker and J. Vennekens. The well-founded semantics is the principle of inductive definition, revisited. In C. Baral, G. De Giacomo, and T. 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.

M. Denecker, V. Marek, and M. Truszczyński. Fixpoint 3-valued semantics for autoepistemic logic. In J. Mostow and C. Rich, editors, *AAAI'98*, pages 840–845, Madison, Wisconsin, July 26-30 1998. MIT Press. ISBN 0-262-51098-7. URL http://www.aaai.org/Papers/AAAI/1998/AAAI98-119.pdf.

M. Denecker, V. Marek, and M. Truszczyński. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In J. 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.

M. Denecker, M. Bruynooghe, and V. Marek. Logic programming revisited: Logic programs as inductive definitions. *ACM Trans. Comput. Log.*, 2(4): 623–654, 2001.

M. Denecker, V. Marek, and M. 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.

M. Denecker, V. Marek, and M. 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.

M. Denecker, V. Marek, and M. Truszczyński. Reiter's default logic is a logic of autoepistemic reasoning and a good one, too. In G. Brewka, V. Marek, and M. Truszczyński, editors, *Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary*, pages 111–144. College Publications, 2011. URL http://arxiv.org/abs/1108.3278.

M. Denecker, Y. Lierler, M. Truszczyński, and J. Vennekens. A Tarskian informal semantics for answer set programming. In A. Dovier and V. S. Costa, editors, *ICLP (Technical Communications)*, volume 17 of *LIPIcs*, pages 277–289. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. ISBN 978-3-939897-43-9.

H. B. Enderton. *A Mathematical Introduction To Logic*. Academic Press, second edition, 2001.

R. Fagin, J. Y. Halpern, and M. Y. Vardi. A model-theoretic analysis of knowledge. *J. ACM*, 38(2):382–428, 1991. doi: 10.1145/103516.128680. URL http://doi.acm.org/10.1145/103516.128680.

R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995. URL http://library.books24x7.com.libproxy.mit.edu/toc.asp?site=bbbga&#38;bookid=7008.

A. A. Falkner and A. Haselböck. Challenges of knowledge evolution in practice. *AI Communications*, 26(1):3–14, 2013. doi: 10.3233/AIC-120542. URL http://dx.doi.org/10.3233/AIC-120542.

A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. *Knowledge-based Configuration: From Research to Business Cases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2014. ISBN 012415817X, 9780124158177.

M. Fitting. Barcan both ways. *Journal of Applied Non-Classical Logics*, 9 (2-3):329–344, 1999. doi: 10.1080/11663081.1999.10510970. URL http://dx.doi.org/10.1080/11663081.1999.10510970.

G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner. Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems*, 13(4):59–68, 1998. doi: 10.1109/5254.708434. URL http://dx.doi.org/10.1109/5254.708434.

E. Friedman-Hill. *Jess in action: Rule-based systems in Java*. In Action series. Manning, 2003. ISBN 9781930110892. URL http://books.google.be/books?id=-xxjRZhyF0IC.

D. Garg. *Proof Theory for Authorization Logic and Its Application to a Practical File System*. PhD thesis, 2009.

D. Garg and F. Pfenning. Stateful Authorization Logic – Proof Theory and a Case Study. *Journal of Computer Security*, 20(4):353–391, 2012.

M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. 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.

M. Gelfond and V. Lifschitz. Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.

V. Genovese. *Modalities in Access Control: Logics, Proof-theory and Application*. PhD thesis, 2012.

P. J. Gmytrasiewicz and E. H. Durfee. A logic of knowledge and belief for recursive modeling: A preliminary report. In W. R. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992.*, pages 628–634. AAAI Press / The MIT Press,

1992. ISBN 0-262-51063-4. URL http://www.aaai.org/Library/AAAI/1992/aaai92-097.php.

Y. Gurevich and I. Neeman. DKAL: Distributed-knowledge authorization language. In *Proceedings of the 2008 21st IEEE Computer Security Foundations Symposium*, pages 149–162. IEEE, 2008.

T. Hadzic. A BDD-based approach to interactive configuration. In M. Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *LNCS*, page 797. Springer, 2004. ISBN 3-540-23241-9. doi: 10.1007/978-3-540-30201-8_76. URL http://dx.doi.org/10.1007/978-3-540-30201-8_76.

T. Hadzic and H. R. Andersen. Interactive reconfiguration in power supply restoration. In P. van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 767–771. Springer, 2005. ISBN 3-540-29238-1. doi: 10.1007/11564751_61. URL http://dx.doi.org/10.1007/11564751_61.

Å. Hagström, S. Jajodia, F. Parisi-Presicce, and D. Wijesekera. Revocations – a classification. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, pages 44–58. IEEE, 2001.

J. Y. Halpern and G. Lakemeyer. Multi-agent only knowing. *J. Log. Comput.*, 11(1):41–70, 2001. doi: 10.1093/logcom/11.1.41. URL http://dx.doi.org/10.1093/logcom/11.1.41.

J. Y. Halpern and Y. Moses. Towards a theory of knowledge and ignorance: Preliminary report. In K. R. Apt, editor, *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI Series*, pages 459–476. Springer Berlin Heidelberg, 1985. ISBN 978-3-642-82455-5. doi: 10.1007/978-3-642-82453-1_16. URL http://dx.doi.org/10.1007/978-3-642-82453-1_16.

F. Heras, A. Morgado, and J. Marques-Silva. Core-guided binary search algorithms for maximum satisfiability. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, 2011. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3713.

L. Hotz, T. Krebs, S. Deelstra, M. Sinnema, and J. Nijhuis. *Configuration in industrial product families - the ConIPF methodology.* IOS Press, Inc., 2006. ISBN 978-3-89838-067-6.

G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic.* Psychology Press, 1996.

S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001. URL `http://portal.acm.org/citation.cfm?id=383891.383894`.

M. Janota. Do SAT solvers make good configurators? In S. Thiel and K. Pohl, editors, *Software Product Lines, 12th International Conference, SPLC 2008, Limerick, Ireland, September 8-12, 2008, Proceedings. Second Volume (Workshops)*, pages 191–195. Lero Int. Science Centre, University of Limerick, Ireland, 2008. ISBN 978-1-905952-06-9.

J. Jansen, I. Dasseville, J. Devriendt, and G. Janssens. Experimental evaluation of a state-of-the-art grounder. In O. Chitil, A. King, and O. Danvy, editors, *Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, Kent, Canterbury, United Kingdom, September 8-10, 2014*, pages 249–258. ACM, 2014. ISBN 978-1-4503-2947-7. doi: 10.1145/2643135.2643149. URL `http://doi.acm.org/10.1145/2643135.2643149`.

I. JRules. Leading the way in business rule management systems. *ILOG, Mar*, 2005.

U. Junker. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 167–172. AAAI Press / The MIT Press, 2004. ISBN 0-262-51183-5. URL `http://www.aaai.org/Library/AAAI/2004/aaai04-027.php`.

U. Junker and D. Mailharro. Preference programming: Advanced problem solving for configuration. *AI EDAM*, 17(1):13–29, 2003. doi: 10.1017/S089006040317103X. URL `http://dx.doi.org/10.1017/S089006040317103X`.

K. Kang. *Feature-oriented Domain Analysis (FODA): Feasibility Study ; Technical Report CMU/SEI-90-TR-21 - ESD-90-TR-222.* Software Engineering Inst., Carnegie Mellon Univ., 1990. URL `https://books.google.be/books?id=yYi5PgAACAAJ`.

A. S. Karatas, H. Oguztüzün, and A. H. Dogru. Mapping extended feature models to constraint logic programming over finite domains. In J. Bosch and J. Lee, editors, *Software Product Lines: Going Beyond - 14th International Conference, SPLC 2010, Jeju Island, South Korea, September 13-17, 2010.*

Proceedings, volume 6287 of *Lecture Notes in Computer Science*, pages 286–299. Springer, 2010. ISBN 978-3-642-15578-9. doi: 10.1007/978-3-642-15579-6_20. URL http://dx.doi.org/10.1007/978-3-642-15579-6_20.

Z. Kiziltan, P. Flener, and B. Hnich. Towards inferring labelling heuristics for CSP application domains. In F. Baader, G. Brewka, and T. Eiter, editors, *KI 2001: Advances in Artificial Intelligence, Joint German/Austrian Conference on AI, Vienna, Austria, September 19-21, 2001, Proceedings*, volume 2174 of *LNCS*, pages 275–289. Springer, 2001. ISBN 3-540-42612-4. doi: 10.1007/3-540-45422-5_20. URL http://dx.doi.org/10.1007/3-540-45422-5_20.

S. C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938. ISSN 00224812. URL http://www.jstor.org/stable/2267778.

S. C. Kleene. *Introduction to Metamathematics.* Van Nostrand, 1952.

K. Konolige. On the relation between default and autoepistemic logic. *Artif. Intell.*, 35(3):343–382, 1988. doi: 10.1016/0004-3702(88)90021-5. URL http://dx.doi.org/10.1016/0004-3702(88)90021-5.

H. J. Levesque. All I know: A study in autoepistemic logic. *Artif. Intell.*, 42(2-3):263–309, 1990. doi: 10.1016/0004-3702(90)90056-6. URL http://dx.doi.org/10.1016/0004-3702(90)90056-6.

B. Li, L. Chen, Z. Huang, and Y. Zhong. Product configuration optimization using a multiobjective genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 30(1):20–29, 2005. ISSN 1433-3015. doi: 10.1007/s00170-005-0035-8. URL http://dx.doi.org/10.1007/s00170-005-0035-8.

L. Libkin. *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

C. Lutz. Complexity and succinctness of public announcement logic. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 137–143. ACM, 2006. ISBN 1-59593-303-4. doi: 10.1145/1160633.1160657. URL http://doi.acm.org/10.1145/1160633.1160657.

I. Lynce and J. P. M. Silva. On computing minimum unsatisfiable cores. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*, 2004.

J. Marques-Silva and J. Planes. Algorithms for maximum satisfiability using unsatisfiable cores. In *Design, Automation and Test in Europe, DATE 2008, Munich, Germany, March 10-14, 2008*, pages 408–413, 2008. doi: 10.1109/DATE.2008.4484715. URL http://dx.doi.org/10.1109/DATE.2008.4484715.

J. P. McDermott. R1: A rule-based configurer of computer systems. *Artif. Intell.*, 19(1):39–88, 1982. doi: 10.1016/0004-3702(82)90021-2. URL http://dx.doi.org/10.1016/0004-3702(82)90021-2.

D. L. McGuinness and J. R. Wright. An industrial-strength description-logics-based configurator platform. *IEEE Intelligent Systems*, 13(4):69–77, 1998. doi: 10.1109/5254.708435. URL http://dx.doi.org/10.1109/5254.708435.

D. G. Mitchell and E. Ternovska. A framework for representing and solving NP search problems. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 430–435. AAAI Press / The MIT Press, 2005. ISBN 1-57735-236-X. URL http://www.aaai.org/Library/AAAI/2005/aaai05-068.php.

S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In T. Dieterich and W. Swartout, editors, *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, July 29 - August 3, 1990, 2 Volumes.*, pages 25–32. AAAI/MIT Press, 1990. ISBN 978-0-262-51057-8. URL http://www.aaai.org/Library/AAAI/1990/aaai90-004.php.

S. Mittal and F. Frayman. Towards a generic model of configuraton tasks. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 1395–1401. Morgan Kaufmann, 1989. ISBN 1-55860-094-9.

R. C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the Workshop on Non-Monotonic Reasoning*, pages 344–354, 1984. URL http://www.sri.com/sites/default/files/uploads/publications/pdf/616.pdf. Reprinted in: M. Ginsberg, ed., *Readings on Nonmonotonic Reasoning*, pages 137–142, Morgan Kaufmann, 1990.

R. C. Moore. Semantical considerations on nonmonotonic logic. *Artif. Intell.*, 25(1):75–94, 1985. doi: 10.1016/0004-3702(85)90042-6. URL http://dx.doi.org/10.1016/0004-3702(85)90042-6.

I. Niemelä. Constructive tightly grounded autoepistemic reasoning. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*, pages 399–405. Morgan Kaufmann, 1991. ISBN 1-55860-160-0.

A. Oberschelp. Untersuchungen zur mehrsortigen quantorenlogik. *Mathematische Annalen*, 145(4):297–333, 1962.

F. T. Piller, T. Harzer, C. Ihl, and F. Salvador. Strategic capabilities of mass customization based e-commerce: Construct development and empirical test. In *47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, USA, January 6-9, 2014*, pages 3255–3264. IEEE, 2014. doi: 10.1109/HICSS.2014.403. URL http://dx.doi.org/10.1109/HICSS.2014.403.

E. Pontelli and T. C. Son. *Justifications* for logic programs under answer set semantics. In S. Etalle and M. Truszczyński, editors, *ICLP*, volume 4079 of *LNCS*, pages 196–210. Springer, 2006. ISBN 3-540-36635-0.

QML. Qml. http://qmlbook.org/, 2015.

R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1): 57–95, 1987. doi: 10.1016/0004-3702(87)90062-2. URL http://dx.doi.org/10.1016/0004-3702(87)90062-2.

R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, 2001. ISBN 9780262264310. URL http://books.google.be/books?id=exa4f6BOZdYC.

D. Schneeweiss and P. Hofstedt. Fdconfig: A constraint-based interactive product configurator. In H. Tompits, S. Abreu, J. Oetsch, J. Pührer, D. Seipel, M. Umeda, and A. Wolf, editors, *Applications of Declarative Programming and Knowledge Management - 19th International Conference, INAP 2011, and 25th Workshop on Logic Programming, WLP 2011, Vienna, Austria, September 28-30, 2011, Revised Selected Papers*, volume 7773 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 2011. ISBN 978-3-642-41523-4. doi: 10.1007/978-3-642-41524-1_13. URL http://dx.doi.org/10.1007/978-3-642-41524-1_13.

M. Shanahan. *Solving the Frame Problem - a Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press, 1997. URL http://mitpress.mit.edu/books/solving-frame-problem.

K. M. Shchekotykhin, G. Friedrich, P. Rodler, and P. Fleiss. Interactive ontology debugging using direct diagnosis. In P. Lambrix, G. Qi, M. Horridge, and B. Parsia, editors, *Proceedings of the Third International Workshop on Debugging Ontologies and Ontology Mappings, WoDOOM 2014, co-located with 11th Extended Semantic Web Conference (ESWC 2014), Anissaras/Hersonissou, Greece, May 26, 2014.*, volume 1162 of *CEUR Workshop Proceedings*, pages 39–50. CEUR-WS.org, 2014. URL http://ceur-ws.org/Vol-1162/paper4.pdf.

I. Shlyakhter, R. Seater, D. Jackson, M. Sridharan, and M. Taghdiri. Debugging overconstrained declarative models using unsatisfiable cores. In *ASE*, pages 94–105. IEEE Computer Society, 2003. ISBN 0-7695-2035-9.

T. Syrjänen. Debugging inconsistent answer set programs. In J. Dix and A. Hunter, editors, *Proceedings of the Eleventh International Workshop on Non-Monotonic Reasoning, NMR 2006, Lake District, UK, 30 May - 1 June*, pages 77–84, 2006.

J. Tiihonen, M. Heiskala, A. Anderson, and T. Soininen. Wecotin - A practical logic-based sales configurator. *AI Commun.*, 26(1):99–131, 2013. doi: 10.3233/AIC-2012-0547. URL http://dx.doi.org/10.3233/AIC-2012-0547.

J. Van den Bussche and J. Paredaens. The expressive power of complex values in object-based data models. *Information and Computation*, 120:220–236, 1995.

A. Van Gelder. The alternating fixpoint of logic programs with negation. *J. Comput. Syst. Sci.*, 47(1):185–221, 1993.

P. Van Hertum. Combining logic and business rule systems. In *Web Reasoning and Rule Systems: 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings*, volume 8741, page 253. Springer, 2014.

P. Van Hertum, J. Vennekens, B. Bogaerts, J. Devriendt, and M. Denecker. The effects of buying a new car: An extension of the IDP knowledge base system. *CW Reports*, CW640, 2013a.

P. Van Hertum, J. Vennekens, B. Bogaerts, J. Devriendt, and M. Denecker. The effects of buying a new car: An extension of the IDP knowledge base system. *TPLP*, 13(4–5-Online-Supplement), 2013b.

P. Van Hertum, M. Cramer, B. Bogaerts, and M. Denecker. Distributed autoepistemic logic and its application to access control. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1286–1292. IJCAI/AAAI Press, 2016a. ISBN 978-1-57735-770-4. URL http://www.ijcai.org/Abstract/16/186.

P. Van Hertum, I. Dasseville, G. Janssens, and M. Denecker. The KB paradigm and its application to interactive configuration. In M. Gavanelli and J. H. Reppy, editors, *Practical Aspects of Declarative Languages - 18th International Symposium, PADL 2016, St. Petersburg, FL, USA, January 18-19, 2016. Proceedings*, volume 9585 of *Lecture Notes in Computer Science*, pages 13–29. Springer, 2016b. ISBN 978-3-319-28227-5. doi: 10.1007/978-3-319-28228-2_2. URL http://dx.doi.org/10.1007/978-3-319-28228-2_2.

P. Van Hertum, I. Dasseville, G. Janssens, and M. Denecker. The kb paradigm and its application to interactive configuration. *Theory and Practice of Logic Programming*, FirstView:1–27, 7 2016c. ISSN 1475-3081. doi: 10.1017/S1471068416000156. URL http://journals.cambridge.org/article_S1471068416000156.

M. Vanden Bossche, P. Ross, I. MacLarty, B. Van Nuffelen, and N. Pelov. Ontology driven software engineering for real life applications. In *3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE)*, 2007.

J. Vennekens, D. Gilis, and M. Denecker. Erratum to splitting an operator: Algebraic modularity results for logics with fixpoint semantics (vol 7, pg 765, 2006), Jan. 2007. URL https://lirias.kuleuven.be/handle/123456789/124415.

J. Vennekens, M. Denecker, and M. Bruynooghe. CP-logic: A language of causal probabilistic events and its relation to logic programming. *TPLP*, 9 (3):245–308, 2009. doi: 10.1017/S1471068409003767. URL https://lirias.kuleuven.be/handle/123456789/229813.

H. Vlaeminck, J. Vennekens, and M. Denecker. A logical framework for configuration software. In A. Porto and F. J. López-Fraguas, editors, *Proceedings of the 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, September 7-9, 2009, Coimbra, Portugal*, pages 141–148. ACM, 2009. ISBN 978-1-60558-568-0.

H. Vlaeminck, J. Vennekens, M. Bruynooghe, and M. Denecker. Ordered Epistemic Logic: Semantics, complexity and applications. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Knowledge Representation and Reasoning, Rome, 10-14 July 2012*, pages 369–379. AAAI Press, 2012. URL http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4513.

J. Wittocx, M. Mariën, and M. Denecker. The IDP system: A model expansion system for an extension of classical logic. In M. Denecker, editor, *LaSh*, pages 153–165. ACCO, 2008.

J. Wittocx, H. Vlaeminck, and M. Denecker. Debugging for model expansion. In P. M. Hill and D. S. Warren, editors, *ICLP*, volume 5649 of *LNCS*, pages 296–311. Springer, 2009. ISBN 978-3-642-02845-8.

Q. Yang and M. Wooldridge, editors. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos*

Aires, Argentina, July 25-31, 2015, 2015. AAAI Press. ISBN 978-1-57735-738-4.

J. Zhang, S. Li, and S. Shen. Extracting minimum unsatisfiable cores with a greedy genetic algorithm. In *AI 2006: Advances in Artificial Intelligence, 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006, Proceedings*, pages 847–856, 2006. doi: 10.1007/11941439_89. URL http://dx.doi.org/10.1007/11941439_89.

Curriculum Vitae

Pieter Van Hertum was born in Neerpelt, Belgium on December 21, 1989. He studied Techniek-Wetenschappen at Don Bosco College in Hechtel, Belgium. Afterwards, he studied Mathematics at KU Leuven, starting in 2007. He wrote a master's thesis under supervision of prof. dr. Marc Denecker with the title "The simplification of First-order BDD's using Presburger Arithmetic" and graduated Cum Laude in 2012.

After graduating at KU Leuven, Pieter joined the DTAI ("Declarative Languages and Artificial Intelligence") research group to pursue a PhD under supervision of prof. dr. Marc Denecker to research applications of the Knowledge Base Paradigm in Business contexts. In 2013, prof. dr. ir. Gerda Janssens became his co-supervisor and he got involved in the SIEP project, studying in collaboration with researchers at the University of Luxemburg how the Knowledge Base Paradigm can be applied for Access Control.

List of Publications

All publications are available on http://www.cs.kuleuven.be/publicaties/lirias/mypubs.php?unum=U0074117.

Journal and Book Articles

- P. Van Hertum, J. Vennekens, B. Bogaerts, J. Devriendt and M. Denecker. "The effects of buying a new car: An extension of the IDP Knowledge Base System". In: Theory and Practice of Logic Programming, volume 10, (4-5, Online Supplement), 2013b.
- P. Van Hertum, I. Dasseville, G. Janssens, M. Denecker: The KB Paradigm and its application to interactive configuration. Accepted for publication in Theory and Practice of Logic Programming, CoRR abs/1605.01846.

Peer-reviewed Articles at Conferences and Workshops

- M. Cramer, D. Ambrossio and P. Van Hertum: A logic of trust for reasoning about delegation and revocation. In Proceedings of the 20th

ACM Symposium on Access Control Models and Technologies (SACMAT) 2015, Vienna, Austria, 1-3 June 2015. pages 173-184

- P. Van Hertum, I. Dasseville, G. Janssens and M. Denecker. The KB Paradigm and its application to interactive configuration. In Marco Gavanelli and John H. Reppy (eds.), Practical Aspects of Declarative Languages - 18th International Symposium, (PADL) 2016, St. Petersburg, FL, USA, 18-19 January 2016. pages 13-29

- P. Van Hertum, M. Cramer, B. Bogaerts and M. Denecker. "Distributed autoepistemic logic and its application to access control". In Subbarao Kambhampati (ed.), Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI) 2016, New York, NY, USA, 9-15 July 2016. pages 1286-1292.

Peer-reviewed Abstracts

- P. Van Hertum. "Combining logic and business rules". In: Kontchakov, Roman, Mugnier, Marie-Laure (Eds.), Proceedings of the 8th International Conference on Web Reasoning and Rule Systems (RR) 2014, Athens, Greece, 15-17 September 2014. Lecture Notes in Computer Science: 2014-09, Volume: 8741, Pages: 253-254

Informal and Other Publications

- P. Van Hertum, J. Vennekens, B. Bogaerts, J. Devriendt and M. Denecker. "The effects of buying a new car: An extension of the IDP Knowledge Base System". In: CW Reports, CW640 2013a.

- M. Cramer, P. Van Hertum, D. Ambrossio and M. Denecker. "Modelling delegation and revocation Schemes in IDP". In CoRR abs/1405.1584 (2014)

Posters and Presentations at Miscellaneous Events

- Presentation: "The effects of buying a new car: An extension of the IDP knowledge base system". International Conference on Logic Programming 2013, Istanbul, Turkey, 24 - 29 August 2013.

- Poster & Presentation: "Combining logic and business rules". International Conference on Web Reasoning and Rule Systems (RR) 2014, Athens, Greece, 15-17 September 2014.
- Presentation: Advantages, challenges and applications of the Knowledge Base System paradigm, DTAI Seminar October 2014, KU Leuven.
- Presentation: “Using approximation fixpoint theory for new semantics with multi-agent autoepistemic logic with full introspection” Workshop SIEP Project, October 2015, University of Luxemburg.
- Presentation: Reasoning about knowledge: modal logics for different kinds of applications, DTAI Seminar 2 May 2016, KU Leuven.
- Poster & Presentation : "Distributed autoepistemic logic and its application to access control". International Joint Conference on Artificial Intelligence (IJCAI) 2016, New York, NY, USA, 9-15 July 2016.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DECLARATIEVE TALEN EN ARTIFICIËLE INTELLIGENTIE
Celestijnenlaan 200A box 2402
B-3001 Leuven
pieter.vanhertum@cs.kuleuven.be