Nederlandse Samenvatting

Algebraïsche en Logische Studie van Constructieve Processen in Kennisrepresentatie

Joost Vennekens

Samenvatting

Het onderwerp van deze thesis zijn constructieve processen en hun rol in kennisrepresentatie. In een eerste deel van de tekst, bestuderen we de eigenschappen van dergelijke processen op een algebraïsche manier, binnen het kader van *approximatie theorie*. De abstracte resultaten die we hier bekomen, laten ons toe om op een uniforme en vrij eenvoudige manier concrete stellingen voor verschillende kennisrepresentatie-talen af te leiden. Met name bestuderen we op deze manier de onderwerpen van *modulariteit* en *predikaat introductie* voor logische programma's, auto-epistemische logica en default logica. In een tweede deel van de tekst, onderzoeken we het verband tussen constructieve processen en het concept van causaliteit. Door middel van een analyse van een bepaald soort causale uitspraken, tonen we aan dat causaliteit een inherent dynamisch aspect heeft, dat op een natuurlijke manier aanleiding geeft tot een formalizatie in termen van probabilistische processen. We vergelijken de resulterende taal met Bayesiaanse netwerken en tonen aan dat er een nauw verband bestaat met logische programma's, waaruit volgt dat we ook uitdrukking in dit formalisme op een causale manier kunnen interpreteren.

1 Inleiding

1.1 Kennisrepresentatie en constructieve processes

Als we kennis willen delen, bewaren, of misschien zelfs alleen al maar denken, moeten we deze uitdrukking in taal. Het is bekend dat uitdrukkingen in natuurlijke talen (Nederlands, Frans, ...) vaak vaag of ambigu zijn. In het dagelijkse leven stelt dit weinig of geen problemen, aangezien een goede verstaander de bedoelde betekenis meestal wel zal begrijpen en ontbrekende informatie vanuit zijn eigen kennis kan aanvullen. In wiskunde of exacte wetenschappen ligt de situatie echter anders. Hier is precisie en eenduidigheid vaak van een dermate groot belang, dat natuurlijke taal eenvoudigweg niet meer voldoet. Wiskundigen maken in dergelijke situaties dan ook gebruik van hun eigen, symbolische talen, zoals bijvoorbeeld wiskundige eerste-orde logica. Het belangrijkste onderscheid tussen dergelijke formele talen en een informele natuurlijk taal, is dat een formele taal de betekenis van elke uitdrukking exact en eenduidig definieert.

Formele talen zijn ook in de informatica van groot belang. Kennis die exact en eenduidig geformuleerd is, is immers ook kennis die algoritmisch gemanipuleerd kan worden en dus voor een computer bruikbaar is. Dit schept de mogelijkheid tot *declaratief* programmeren, waarbij een programmeur zich niet meer hoeft te bekommeren om *hoe* iets berekend kan worden, maar enkel maar moet aangeven *wat* hij wilt weten. Als hij dit in een formele taal kan neerschijven, samen met voldoende achtergrondkennis over het probleem in kwestie, dan kan—in het ideale geval—het gewenste resultaat gevonden worden door het toepassen van algemene redeneermethodes op dit specifieke geval. Opdat deze aanpak zinvol zou zijn, is het natuurlijk van belang dat de gebruikte formele taal ook effectief geschikt is om de relevante kennis in voor te stellen. Omdat het hier vaak om andere soorten van kennis gaat dan strikt wiskundige, en er in deze context bovendien ook andere eisen aan een taal gesteld worden dan in de wiskunde (zoals bijvoorbeeld berekenbaarheid), is er hier nood aan andere talen nodig dan enkel maar wiskundige eerste-orde logica.

De studie van het ontwerp en gebruik van dergelijke talen is het onderwerp van het domein van de *kennisrepresentatie*. Constructieve processen spelen een belangrijke rol in dit domein. Veel van de talen die hier beschouwd worden maken immers gebruik van dergelijke processen om de betekenis van uitdrukkingen te definiëren. Een voorbeeld hiervan is de taal *ID-logica* (Denecker en Ternovska 2004), die gebruik maakt van logische programma's om het wiskundige concept van een inductieve definitie te formalizeren. Een dergelijke definitie bestaat in wezen uit een aantal basisgevallen, aangevuld met een "recept" dat aangeeft hoe, uit het feit dat bepaalde objecten reeds tot het gedefinieerde concept behoren, de aanwezigheid van andere objecten kan worden afgeleid. Een typisch voorbeeld is volgende definitie van "bereikbaarheid" (bijvoorbeeld: is het mogelijk om via het openbaar vervoer vanuit een bepaald vertrekpunt tot op een bepaalde bestemming te raken?). Het basisgeval is hier dat x bereikbaar is vanuit y indien er een directe verbinding van y naar x bestaat. Het inductieve "recept" zegt dan: als x bereikbaar is uit y en y bereikbaar is uit z, dan is x ook bereikbaar uit z. ID-logica is een formele taal, waarin dergelijke definities op een natuurlijke manier kunnen worden neergeschreven. In het bovenstaande geval, schrijven we bijvoorbeeld:

$$\left\{\begin{array}{l} \forall x, y\ Bereikbaar(x,y) \leftarrow Verbinding(y,x). \\ \forall x, z\ Bereikbaar(x,z) \leftarrow \exists y\ Bereikbaar(x,y) \wedge Bereikbaar(y,z). \end{array}\right\}$$

Gegeven een dergelijke definitie is het dan mogelijk om concrete vragen te beantwoorden, zoals bijvoorbeeld: "Is Westerlo bereikbaar vanuit Leuven?" Om dit te doen, volstaat het om het inductieve recept herhaaldelijk toe te passen, vertrekkend van het basisgeval, totdat er genoeg van het gedefinieerde concept geconstrueerd is om het gewenste antwoord te kunnen geven. Het is dit soort van procedure die met de term "constructief proces" bedoeld wordt. Dergelijke processen spelen niet alleen in ID-logica een belangrijke rol, maar komen ook in andere talen (zoals logische programma's, auto-epistemische logica, default logica, ...) aan bod.

1.2 Inhoud van deze thesis

In deze thesis bestuderen we constructieve processen in twee verschillende omstandigheden.

In een eerste deel zullen we dergelijke processen op een abstracte manier bestuderen, binnen het kader van *approximatie theorie* (Denecker, Marek, en Truszczynski 2003). Dit is een algebraïsche theorie die op een natuurlijk manier al de voornaamste semantieken van een aantal verschillende kennisrepresentatietalen vangt, waaronder logische programma's, auto-epistemische logica en default logica. Dit betekent dat we door één enkele analyse op het niveau van approximatie theorie meteen over al deze talen iets te weten te komen. In deze thesis bestuderen we twee onderwerpen op deze manier: modulariteit en predikaat introductie. Voor deze beide onderwerpen zullen we aantonen dat onze abstracte resultaten binnen approximatie theorie ons toelaten om op een uniforme en vrij eenvoudige manier concrete stellingen af te leiden, die een aantal reeds bekende resultaten voor verschillende talen omvatten of veralgemenen.

In het tweede deel van deze thesis bestuderen we dan het verband tussen constructieve processen en het concept van *causaliteit*. We vertrekken hier van een analyse van een bepaald soort van causale uitspraken en tonen aan dat deze een inherent dynamisch aspect hebben, dat op een natuurlijke manier aanleiding geeft tot constructieve processen. Omdat causale uitspraken vaak gepaard gaan met een zekere mate van onzekerheid, zullen we in deze context probabilistische processen beschouwen, waarvan de basisstappen een niet-deterministisch effect kunnen hebben. De resulterende probabilistische modelleringstaal vergelijken we met die van Bayesiaanse netwerken. Bovendien tonen we aan dat er een nauw verband bestaat met logische programma's, wat aantoont dat ook uitdrukkingen in dit formalisme op een causale manier geïnterpreteerd kunnen worden.

2 Algebraïsche studie van talen met een vastepuntssemantiek

In deze sectie lichten we het eerste deel van deze thesis toe, waarin we constructieve processen bestuderen binnen het algebraïsche kader van approximatie theorie.

2.1 Vooraf: Approximatie theorie

In deze voorafgaande sectie vatten we kort enkele kernbegrippen uit approximatie theorie samen, zoals deze in (Denecker, Marek, en Truszczynski 2003) worden voorgesteld.

Constructieve processen kunnen op een abstracte, wiskundige manier bestudeerd worden. Laten we illustreren hoe dit voor de inductieve definitie van het concept "bereikbaarheid" in zijn werk zou kunnen gaan. We beschouwen hiervoor de grafe in Figuur 1. Het concept "bereikbaarheid" is van toepassing op paren van knooppunten in dit netwerk; bijvoorbeeld, het paar $(Leuven, Westerlo)$

Figuur 1: Grafe ter illustratie van "bereikbaarheid".

is er eentje dat tot dit concept zou moeten behoren (*Leuven* is immers bereikbaar vanuit $Westerlo$), terwijl dit voor het paar $(Westerlo, Roosdaal)$ niet zo is.

We kunnen nu functies beschouwen, die een verzameling van dergelijke paren afbeelden op een nieuwe verzameling van paren. Laat ons concreet kijken naar de functie O die een verzameling V van paren van knooppunten afbeeldt op de verzameling V', die precies alle paren (x, y) bevat, waarvoor:

- Ofwel (x, y) een verbinding in het netwerk is;
- Ofwel er een z bestaat, zodanig dat zowel (x, z) als (z, y) tot V behoren.

Bijvoorbeeld, als we deze O toepassen op de verzameling

$$V = \{(Westerlo, Aarschot), (Aarschot, Leuven)\},$$

dan krijgen we

$$O(V) = \{(Westerlo, Aarschot), (Aarschot, Leuven), (Westerlo, Leuven)\}.$$

Deze functie O heeft nu de eigenschap dat ze *monotoon* is, dwz. dat indien V een verzameling is die alle paden bevat die ook in verzameling W zitten ($V \supseteq W$), dan zal $O(V)$ ook alle paden bevatten die in $O(W)$ zitten ($O(V) \supseteq O(W)$). Zoals aangetoond werd door Tarski, heeft een dergelijke monotone functie O een interessante eigenschap: indien we O herhaaldelijk toepassen op de lege verzameling, dan bekomen we een stijgende reeks, die uiteindelijk een limiet zal bereiken, en deze limiet is tevens de kleinste verzameling V waarvoor het zo is dat het toepassen van O op V geen bijkomende paden meer zal toevoegen ($V \supseteq O(V)$); het is zelfs zo dat deze limiet een vast punt is van O, dwz. dat $O(V) = V$, en dat hij bovendien het kleinste vaste punt van O is. Het is nu precies deze limiet die correct de betekenis van het inductief gedefinieerde concept "bereikbaarheid" vat. Inderdaad, in het geval van ons voorbeeld, levert deze reeks de volgende limiet op:

$$\begin{aligned}\{\} &\mapsto \{ber(A, W), ber(L, A), ber(L, B), ber(B, R)\} \\ &\mapsto \{ber(A, W), ber(L, A), ber(L, B), ber(B, R), ber(L, W), ber(L, R)\}.\end{aligned}$$

Op deze manier kan de theorie van Tarski ons dus helpen bij het begrijpen van inductieve definities. Approximatie theorie is nu in wezen niets anders dan een veralgemening van Tarski's theorie naar ook het niet-monotone geval. Concreet wordt er in deze theorie een constructie opgezet, die ook het gewenste resultaat oplevert voor functies O waarvoor er verzamelingen $V \supseteq W$ zijn zodat

$O(V) \not\supseteq O(W)$. Dit gebeurt door het definiëren van het *gefundeerde vaste punt* en de *stabiele vaste punten* van een dergelijke operator.

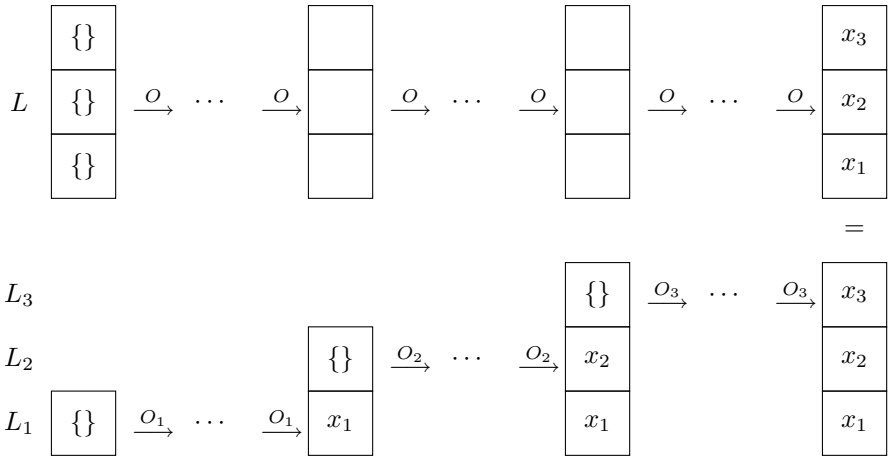
Een typische voorbeeld van een inductieve definitie die aanleiding geeft tot een niet-monotone functie O, vinden we terug in klassieke propositionele logica. Hier wordt de waarheid van een bepaalde formule φ gedefinieerd aan de hand van een inductie over de logische connectieven "en", "of", "niet". Zo zeggen we bijvoorbeeld dat een formule van de vorm "φ en ψ" waar is als zowel φ waar is en ψ waar is. In het geval van de negatie zeggen we dan "niet φ" waar is als φ onwaar is. We zien hier dus dat als er initieel *meer* formules φ waar zijn, dan zullen er na het toepassen van de bijhorende functie O juist *minder* formules van de vorm "niet φ" waar zijn. Dit gedrag valt buiten het bereik van Tarski's theorie, maar wordt door approximatie theorie wel correct behandeld, in die zin dat het gefundeerde vaste punt van O precies overeenkomt met de gewenste semantiek.

In (Denecker, Marek, en Truszczynski 2003) wordt aangetoond dat de voornaamste semantieken van zowel logische programma's, auto-epistemische logica, als default logica op een natuurlijk manier in het raamwerk van approximatie theorie passen. Door gebruik te maken van deze theorie wordt het dan ook mogelijk om eigenschappen van al deze verschillende talen op een uniforme en algemene manier te bestuderen. In wat volgt, doen we dit voor twee onderwerpen: modulariteitseigenschappen en de transformatie van predikaat introductie.

2.2 Modulariteit

Het eerste onderwerp is dat van *modulariteit*. De centrale vraag is hier de volgende. Veronderstel dat we geïnteresseerd zijn in een bepaald domein, dat we op één of andere manier kunnen opsplitsen in een aantal deelgebieden. Als we nu onze kennis over dit domein willen neerschrijven, zullen we geneigd zijn om in eerste instantie elk van deze deelgebieden individueel te beschouwen en voor elk hiervan een afzonderlijke voorstelling van onze kennis erover te construeren. Een voor de hand liggende vraag is dan of we door het samenvoegen van deze verschillende brokjes kennis een correcte beschrijving van het hele domein zullen verkrijgen. Dit kan natuurlijk enkel maar het geval zijn, indien de opdeling van het domein in deelgebieden op een redelijke manier gebeurd is, zodat er geen belangrijke interacties verloren gaan wanneer de delen enkel maar op zichzelf beschouwd worden. In het eerste deel van deze thesis formalizeren we dit in het kader van approximatie theorie.

Concreet kijken we opnieuw naar een functie O zoals we deze in Sectie 2.1 beschouwd hebben. Het opdelen van het domein in deelgebieden komt nu overeen met het splitsen van de semantische ruimte L waarop deze functie werkzaam is (bijvoorbeeld, de functie O voor ons bereikbaarheidsvoorbeeld werkte op de verzameling van alle paren van knooppunten) in een aantal deelruimten $L_1, L_2, \ldots$ Op elk van deze deelruimten L_i beschouwen we dan een functie O_i, die in wezen niets anders doet dan het gedrag van O op deze specifieke deelruimte nabootsen. De vraag die we ons nu stellen is of we deze verzameling van 'kleine' functies $O_1, \ldots, O_n$ kunnen gebruiken om het gedrag van de 'grote' operator O na te



Figuur 2: Onder gepaste voorwaarden is het mogelijk om het constructie proces van de 'grote' functie O na te bootsen met de 'kleine' functies $O_1, \ldots, O_n$.

bootsen. Concreet betekent dit dat we de volgende twee constructie processen, die in Figuur 2 geïllustreerd worden, met elkaar vergelijken:

- Ofwel voeren we een constructie proces uit met de oorspronkelijke O;
- Ofwel voeren we eerst een constructie proces uit met enkel maar de functie O_1, totdat er een bepaalde limiet x_1 bereikt wordt, waarna we dan op deze limiet voortbouwen door verder te werken met O_2, wat leidt tot een limiet x_2, waarna we dan op deze limiet weer verder bouwen met O_3, en zo voort, tot en met O_n.

In deze thesis bepalen we een aantal voorwaarden waaronder deze twee verschillende constructie methodes hetzelfde resultaat zullen opleveren en de functie O dus wel degelijk kan opgesplitst worden in de verzameling van kleinere functies $O_1, \ldots, O_n$. We tonen aan dat dit geldt voor alle constructie processen van approximatie theorie, d.w.z. ook voor gefundeerde en stabiele vaste punten. Dit resultaat kunnen we dan kunnen gebruiken voor het opsplitsen van theorieën in verschillende kennisrepresentatie-talen.

In de context van logisch programmeren, bewijzen we bijvoorbeeld een stelling die ons toelaat om te besluiten dat we het gefundeerde model van het volgende programma

$$\left\{ \begin{array}{l} P \leftarrow P \wedge R. \\ R \leftarrow \neg S. \end{array} \right\}$$

kunnen berekenen door eerst het gefundeerde model van het programma

$$\{R \leftarrow \neg S\} \tag{1}$$

te berekenen (namelijk: R is waar en S is onwaar) en dit dan samen te voegen met het gefundeerde model van het programma

$$\{P \leftarrow P \wedge \mathbf{waar}\}. \tag{2}$$

In dit laatste programma hebben we het atoom R vervangen door de waarheidswaarde **waar**, die het verkregen had in het gefundeerde model van programma (1).

Het resultaat dat we voor logische programma's bewezen hebben, veralgemeent (althans voor de syntax van normale logische programma's) stellingen uit (Lifschitz en Turner 1994; Eiter, Gottlob, en Mannila 1997; Denecker en Ternovska 2004; Verbaeten, Denecker, en Schreye 2000). Daarnaast hebben we dezelfde stelling uit approximatie theorie ook toegepast op auto-epistemische en default logica, waar we resultaten uit (Gelfond en Przymusinska 1992; Niemelä en Rintanen 1994) en (Turner 1996) veralgemeend hebben.

2.3 Predikaat introductie

Een tweede onderwerp dat we behandelen is dat van *predikaat introductie*. In de context van ID-logica (of logische programma's) wordt hiermee het volgende bedoeld. Veronderstel dat we bijvoorbeeld een definitie van het concept "vierkant" gegeven hebben, die zegt:

$$\begin{array}{c}\text{Een vierkant is een vierhoek met enkel rechte hoeken,}\\ \text{waarvan alle zijden even lang zijn.}\end{array} \tag{3}$$

Het kan dan interessant zijn om uit de voorwaarde "een vierhoek met enkel rechte hoeken en even lange zijden" het concept "een vierhoek met enkel rechte hoeken" te isoleren en hiervoor de nieuwe naam *rechthoek* te introduceren. Dit kan nuttig zijn om de definitie leesbaarder te maken of omdat we dit nieuwe concept *rechthoek* nog op andere plaatsen willen gebruiken. Het is dit soort van transformatie waarnaar de term predikaat introductie verwijst: uit de voorwaarde van één van de regels van een oorspronkelijke definitie selecteren we een deelformule, die we dan vervangen door nieuw predikaat, waarvoor we een bijkomende definitie introduceren. In ons voorbeeld, zouden we de definitie bestaande uit regel (3) omzetten in de definitie bestaande uit de volgende twee regels:

- een vierkant is een rechthoek waarvan alle zijden even lang zijn;
- een rechthoek is een vierhoek met enkel rechte hoeken.

Het is hierbij overigens niet zo dat we noodzakelijk het nieuwe concept rechtstreeks moeten definiëren in termen van de vervangen deelformule. In plaats daarvan kan eender welke equivalente definitie gebruikt worden. De vraag die ons uiteindelijk interesseert, is dan natuurlijk onder welke voorwaarden een dergelijke transformatie de betekenis van de oorspronkelijke definitie bewaart.

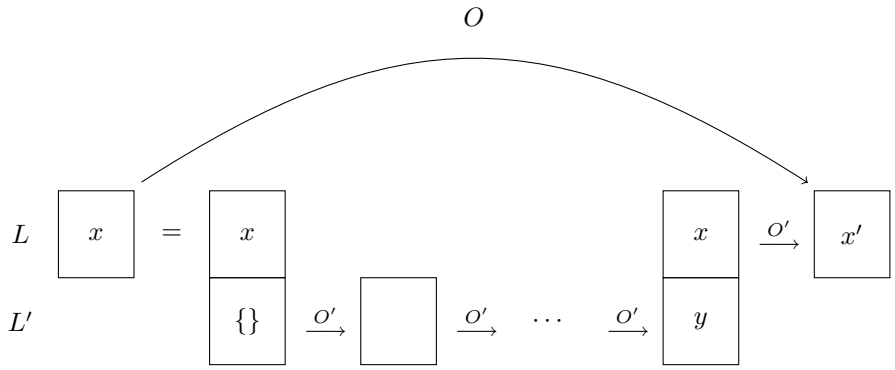
Ook deze vraag kan behandeld worden in het kader van approximatie theorie. Hiertoe introduceren we het concept van een *vaste-puntsextensie* van een operator. De intuïtie hierachter is als volgt. Net zoals in het voorgaande, komt er met onze oorspronkelijke definitie een functie O op een bepaalde semantische ruimte L overeen (in dit geval, alle mogelijke opdelingen van vlakke figuren in een verzameling "vierkanten" en een verzameling "niet-vierkanten"). Het invoeren van een nieuw predikaat betekent nu, algebraïsch gezien, dat we deze ruimte L gaan uitbreiden met een nieuwe ruimte L' (alle mogelijke opdelingen van vlakke figuren in een verzameling "rechthoeken" en een verzameling "niet-rechthoeken"). De betekenis van de nieuwe definite wordt dan ook gegeven door een functie O' op het product $L \times L'$ van de oude en de nieuwe ruimte. We gaan nu op zoek naar voorwaarden waaronder het de limiet van de oorspronkelijke operator O samenvalt met die van de nieuwe operator O' (of, preciezer gezegd, met de beperking hiervan tot de oorspronkelijke ruimte L).

Het kern-idee achter deze voorwaarden is als volgt. Door het toevoegen van een nieuw concept hebben we in wezen een extra indirectie geïntroduceerd. Waar we oorspronkelijk uit het feit dat een vierhoek gelijke zijden en rechte hoeken heeft, onmiddellijk konden afleiden dat hij een vierkant is, moeten we nu immers éérst uit de rechte hoeken afleiden dat hij een rechthoek is, vooraleer we uit de gelijkheid van de zijden kunnen besluiten dat hij een vierkant is. De nieuwe operator O' zal met andere woorden *trager* zijn dan de oorspronkelijke operator O; hij heeft meer stappen nodig om dezelfde conclusies te bereiken. De voorwaarde die we zullen opleggen, bestaat nu essentieel uit het volgende. Als we vertrekken van een interpretatie x voor de oorspronkelijke predikaten, kunnen we hier op twee manier een nieuwe interpretatie x' voor deze predikaten uit afleiden:

- Ofwel passen we de oorspronkelijke functie O toe op x en leiden zo onmiddellijk de nieuwe interpretatie $x' = O(x)$ af;
- Ofwel gebruiken we eerst O' om uit x zoveel mogelijk informatie over het nieuwe predikaat (*rechthoek*) af te leiden en leiden we pas dan, uit zowel de gegeven interpretatie x voor de oorspronkelijke predikaten als de resulterende interpretatie y voor het nieuwe predikaat, een nieuwe interpretatie x' af.

In Figuur 3 worden beide methodes geïllustreerd. Onze voorwaarde waaronder de stabiele en gefundeerde vaste punten van O en O' samenvallen, zal nu, ruw gezegd, erop neerkomen dat beide processen voor elke x dezelfde interpretatie x' moeten construeren.

Een interessante toepassing van dit resultaat is dat we hieruit een manier hebben afgeleid om voorkomens van universele kwantoren in het lichaam van de regels van een logisch programma weg te werken, door hen te vervangen door een recursief gedefiniëerd nieuw predikaat. Concreet betekent dit het volgende. Laten we bijvoorbeeld de volgende definitie van het concept "priemgetal" beschouwen:



Figuur 3: Een vaste-puntsextensie O' van O bereikt uiteindelijk hetzelfde resultaat als O zelf, maar dan trager.

- Een getal n is priem indien voor alle getallen $m < n$ geldt dat n niet deelbaar[1] is door m.

In deze regel komt de uitdrukking "voor alle" voor. Als we deze uitspraak op de voor de hand liggende manier vertalen naar een formula in een logisch programma bekomen we:

$$\forall n\ Priem(n) \leftarrow \forall m\ \ m < n \Rightarrow \neg Deelbaar(n, m).$$

We zien dat de "voor alle m" in de informele beschijving aanleiding geeft tot de "$\forall m$" in deze formula. Regels waarin er iets dergelijks voorkomt, kunnen niet behandeld worden door model generatoren zoals SModels of DLV. Daarom zouden we deze '$\forall$' graag wegwerken. Onder de semantiek van *vervollediging* is er een gekende transformatie (Lloyd en Topor 1984), die hiervoor gebruikt kan worden. Deze transformatie werkt echter niet voor de stabiele of gefundeerde semantiek en kan, als zodanig, in onze context niet gebruikt worden.

Met behulp van onze stelling rond predikaat introductie kunnen dit probleem oplossen. Concreet gaan we de "voor alle" wegwerken door het introduceren van een nieuw predikaat "n heeft geen delers die kleiner zijn dan m":

- Een getal n is priem indien n geen delers heeft die kleiner zijn dan n;

$$\forall n\ Priem(n) \leftarrow GeenDelersKlDan(n, n).$$

- Een getal n heeft geen delers die kleiner zijn dan een getal $m \geq 2$ indien n niet deelbaar is door m en n bovendien geen delers heeft die kleiner zijn dan $m - 1$; voor $m = 1$ geldt dat elk getal n geen delers kleiner dan m

[1] Om het voorbeeld niet nodeloos ingewikkeld te maken, beschouwen we hier het getal 1 niet als een deler.

heeft.

$$\forall n, m\ GeenDelersKlDan(n, m) \leftarrow \neg Deelbaar(n, m) \wedge GeenDelersKlDan(n, m-1).$$
$$\forall n\ GeenDelersKlDan(n, 1).$$

Door dit nieuwe predikaat "n heeft geen delers die kleiner zijn dan m" te definiëren in termen van zichzelf, ontstaat er een soort recursieve lus, die het effect van de oorspronkelijke "voor alle" nabootst. Het resulterende programma kan nu wél dienen als invoer voor, bijvoorbeeld, SModels. In tegenstelling tot de resultaten die voordien bekend waren, is onze stelling rond predikaat introductie algemeen genoeg om de correctheid van deze transformatie aan te tonen.

Naast logische programma's, hebben we dezelfde algebraïsche stelling ook toegepast op auto-epistemische logica. Hier bekomen we een transformatie die, door het introduceren van nieuwe predikaten, genestelde voorkomens van de modale operator K kan wegwerken. Dit biedt een interessant alternatief voor een gelijkaardige transformatie uit (Marek en Truszczyński 1991), aangezien onze transformatie een exponentiële groei in de grootte van de theorie vermijdt door het uitbreiden van het alfabet, terwijl die uit (Marek en Truszczyński 1991) precies het omgekeerde doet.

2.4 Conclusies

In dit eerste deel van deze thesis gebruiken we het raamwerk van approximatie theorie om eigenschappen van kennisrepresentatie talen met een procesgebaseerde semantiek te bestuderen. Het interessante aan deze aanpak is dat we hierdoor kunnen werken op een erg abstract niveau, waarbij we geen rekening moeten houden met de specifieke details van een concrete taal. Als zodanig kunnen er uit één enkele stelling in approximatie theorie vaak een aantal verschillende concrete resultaten worden afgeleid.

Een eerste onderwerp dat we op deze manier bestudeerd hebben is dat van *modulariteit*. Hiervoor ontwikkelden we een algebraïsche theorie rond het opsplitsen van functies, die ons uiteindelijk in staat stelde om een aantal bestaande resultaten (deels) te veralgemenen. Concreet gaat het hier over resultaten voor logische programma's (Lifschitz en Turner 1994; Eiter, Gottlob, en Mannila 1997; Denecker en Ternovska 2004; Verbaeten, Denecker, en Schreye 2000), auto-epistemische logica (Gelfond en Przymusinska 1992; Niemelä en Rintanen 1994) en default logica (Turner 1996).

Een tweede onderwerp was dat van *predikaat introductie*. Ook hiervoor ontwikkelden we een algebraïsche theorie, ditmaal gebaseerd op het concept van een vaste-puntsextensie. Opnieuw stelden we vast dat deze theorie een aantal bestaande resultaten veralgemeent. Voor logische programma's hebben we het hier over werk uit (Van Gelder 1993; Dix en Müller 1994; Aravindan en Dung 1995). Ook in auto-epistemische logica vond onze theorie een interessant toepassing, namelijk het elimineren van genestelde voorkomens van de modale

operator K. Dit biedt een alternatief voor een transformatie uit (Marek en Truszczyński 1991).

De voornaamste bijdrage die in dit eerste deel van dit doctoraat geleverd wordt, is dat we aantonen dat approximatie theorie inderdaad een bruikbaar raamwerk is, waarbinnen eigenschappen van bepaalde kennisrepresentatie talen op een algemene manier bestudeerd kunnen worden. Het zinvolle van deze aanpak blijkt uit het feit dat onze resultaten een groot aantal afzonderlijk bewezen stellingen omvatten en veralgemenen.

3 Een causale probabilistische logica

In het tweede deel van deze tekst onderzoeken we het verband tussen constructieve processes en *causaliteit*. Om onze interesse in dit onderwerp te motiveren, beginnen we met een analyse van de intuïtieve betekenis van causale uitspraken van de volgende vorm:

Een longontstekening veroorzaakt borstpijn. (4)

Er is een voor de hand liggende component aan de betekenis van deze uitspraak, namelijk dat mensen met een longontstekening ook borstpijn hebben. We merken echter ook twee subtielere eigenschappen op.

- De causale uitspraak heeft een *dynamisch* karakter. Ze handelt in wezen over iets dat kan *gebeuren*, over een bepaald mechanisme dat in gang wordt gezet—in dit geval een biologisch proces met virussen, longweefsel en impulsen langs zenuwbanen, dat uiteindelijk resulteert in het fenomeen van pijn. Een dergelijk mechanisme heeft ook een bepaalde tijdsduur, waardoor het kan zijn dat als een patiënt *nu* een longontsteking oploopt, de bijhorende borstpijn zich pas op een *later* tijdstip manifesteert. Dit staat in contrast tot, bijvoorbeeld, de statische uitspraak dat longontsteking borstpijn impliceert, aangezien deze aangeeft dat als we nu een longontsteking observeren, we dan ook nu borstpijn moeten observeren. We zullen de term *gebeurtenis* gebruiken om te verwijzen naar het in werking treden van een dergelijke mechanisme, dus naar het plaatsvinden van, bijvoorbeeld, een dergelijk biologisch proces.

- Zeggen dat longontsteking borstpijn veroorzaakt, impliceert dat borstpijn iets is dat veroorzaakt moet worden. De uitspraak verwijst naar een *a priori* natuurlijke stand van zaken en in deze stand van zaken is het zo dat borstpijn afwezig is. Het is dan enkel maar doordat er hierin veranderingen optreden, doordat bepaalde causale mechanismen geactiveerd worden, dat er toch borstpijn kan komen.

Deze observaties suggereren nu dat als we beschikken over een exhaustieve opsomming van al dergelijke uitspraken die relevant zijn voor een bepaald (aspect van een) domein, dat we dan de uiteindelijke staat van dit domein kunnen

voorspellen door gewoon na te gaan op welke manieren de beschreven gebeurtenissen effectief zouden kunnen plaatsvinden. Laten we bijvoorbeeld de volgende drie uitspraken beschouwen:

Een longontsteking veroorzaakt borstpijn. (5)

Borstpijn veroorzaakt slapeloosheid. (6)

Slapeloosheid veroorzaakt hoofdpijn. (7)

Nu zullen initiëel alle eigenschappen die veroorzaakt kunnen worden (dwz. longontsteking, borstpijn en slapeloosheid) in hun *a priori* staat van afwezigheid zijn. Als de patiënt een longontsteking heeft, dan kan nu de gebeurtenis die beschreven wordt door (5)—en enkel deze—plaatsvinden. Dit verandert dan de staat van het domein en nu heeft de patiënt ook borstpijn. In deze nieuwe staat, kan dan nu de gebeurtenis (6) plaatsvinden, waardoor de patiënt ook hoofdpijn krijgt. Dit leidt dan tot een nieuwe toestand, waarin ook de gebeurtenis beschreven door (7) kan plaatsvinden. Dit brengt ons uiteindelijk in een staat waarin er geen enkele gebeurtenis overblijft die nog kan plaatsvinden, zodat we hier kunnen concluderen dat het domein zijn eindtoestand bereikt heeft.

In de praktijk zal de gebeurtenis die door een dergelijke causale uitspraak beschreven wordt vaak niet deterministisch zijn. In dat geval zeggen we bijvoorbeeld:

Een longontsteking *kan* borstpijn veroorzaken.

Of over een risicovolle operatie zouden we het volgende kunnen zeggen:

De operatie veroorzaakt *ofwel* het herstel *ofwel* de dood van de patiënt.

Het is voor de hand liggend om dergelijke onzekerheid te kwantificeren met kanswaarden, zodat we bijvoorbeeld kunnen zeggen:

Een longontsteking veroorzaakt borstpijn met kans 0.8.

of:

De operatie veroorzaakt met kans 0.7 het herstel
of met kans 0.1 de dood van de patiënt. (8)

Om voor de hand liggende redenen, zullen we naar deze laatste soort van uitspraken verwijzen als *causale probabilistische gebeurtenis-beschrijvingen*, wat we ook wel afkorten tot *CP-gebeurtenis*. Deze uitspraken vormen het centrale onderwerp het tweede deel van deze tekst. Concreet zullen we ons afvragen wat precies de betekenis van een dergelijke uitspraak is en zullen we een kennisrepresentatie taal ontwikkelen, die ons in staat stelt om een domein te modelleren door middel van een opsomming van alle relevante CP-gebeurtenissen.

3.1 De taal CP-logica

In deze sectie definiëren we de taal *CP-logica*, waarin CP-gebeurtenissen formeel kunnen worden voorgesteld. In de formele syntax van deze taal ziet bijvoorbeeld

uitspraak (8) er als volgt uit:

$$(Herstel : 0.7) \vee (Dood : 0.1) \leftarrow Operatie.$$

In het algemeen schrijven een CP-gebeurtenis neer als als een formule van de vorm:

$$(p_1 : \alpha_1) \vee \cdots \vee (p_n : \alpha_n) \leftarrow \varphi,$$

waarbij de p_i gegronde atomen zijn, de α_i kansen zodat $\sum_i \alpha_i \leq 1$, en φ een zin in eerste-orde logica. In eerste instantie beperken we ons voorlopig tot zinnen φ waarin geen negatie voorkomt.

Een eindige verzameling van dergelijke CP-gebeurtenissen noemen we een *CP-theorie*. De betekenis van een dergelijke CP-theorie kunnen we nu ook verklaren aan de hand van constructieve processen, zoals we dit reeds voor de deterministische gebeurtenissen (5), (6) en (7) geïllustreerd hebben. Het enige verschil is dat deze processen nu ook probabilistisch zullen zijn, zodat ze niet meer noodzakelijk leiden naar een unieke eindtoestand, maar ook een probabilistische uitwaaiering van verschillende mogelijkheden kunnen kennen, zoals we nu zullen illustreren.

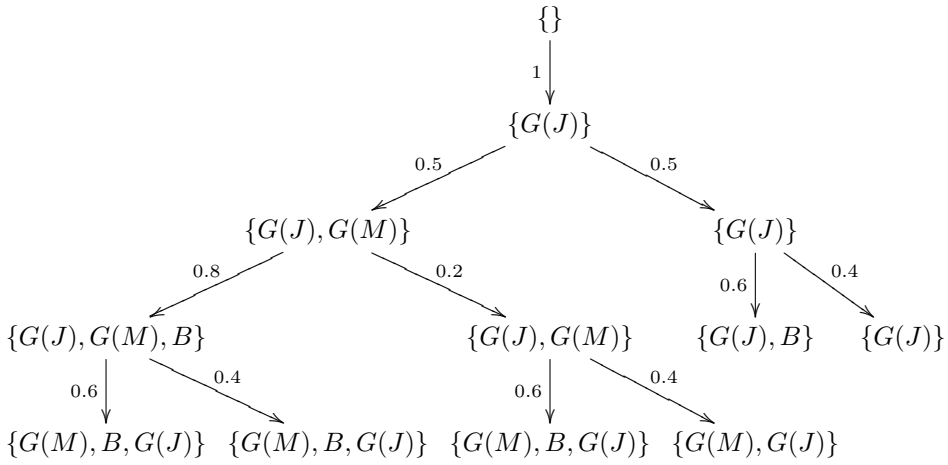
Voorbeeld 1. *Jan en Marie hebben elk een steen in de hand. Jan gooit zijn steen naar een raam. Met kans* 0.5 *gooit ook Marie haar steen naar dit raam. Als Jan zijn steen gooit, breekt deze het raam met kans* 0.6*. Als Marie haar steen gooit, breekt deze het raam met kans* 0.8*. In CP-logica krijgen we de volgende verzameling van CP-gebeurtenissen:*

$$(Gooit(Jan) : 1) \leftarrow . \tag{9}$$
$$(Gooit(Marie) : 0.5) \leftarrow . \tag{10}$$
$$(Gebroken : 0.6) \leftarrow Gooit(Jan). \tag{11}$$
$$(Gebroken : 0.8) \leftarrow Gooit(Marie). \tag{12}$$

Hierbij zijn de eerste twee CP-gebeurtenissen onvoorwaardelijk*, dwz. dat ze steeds zullen plaatsvinden. De eerste CP-gebeurtenis is bovendien* deterministisch*, in de zin dat zij met zekerheid één bepaald effect veroorzaakt.*

Om de betekenis van deze verzameling van CP-gebeurtenissen te bepalen, moeten we kijken naar de processen waartoe deze aanleiding kan geven. Wat er bijvoorbeeld zou kunnen gebeuren, is het volgende:

- Initiëel is elke eigenschap van het domain in zijn natuurlijk toestand, dit wil zeggen, er is nog geen steen gegooid en het raam is niet gebroken.
- In deze toestand kunnen enkel de twee onvoorwaardelijke gebeurtenissen (9) en (10) plaatsvinden. Als bijvoorbeeld de eerste gebeurtenis plaatsvindt (Jan besluit te gooien), dan levert dit met kans 1 de nieuwe toestand $\{Gooit(Jan)\}$ op.



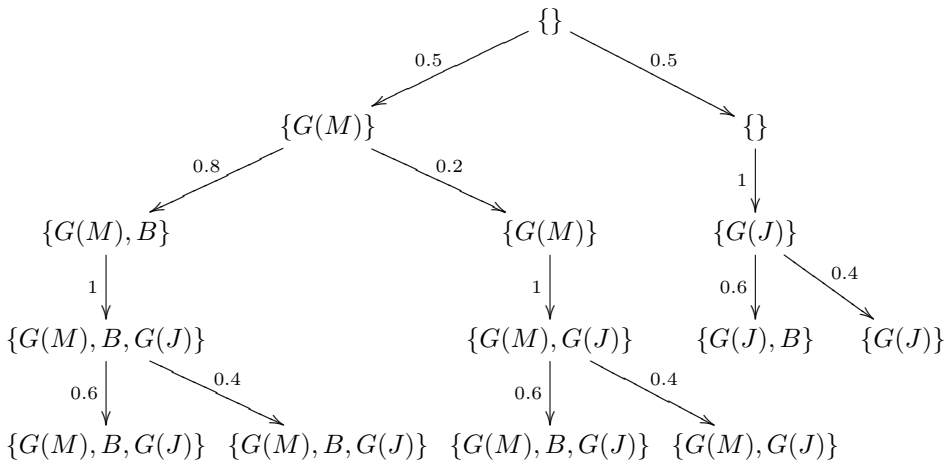
Figuur 4: Een uitvoeringsmodel voor Voorbeeld 1.

- In deze nieuwe toestand kunnen nu gebeurtenissen (10) en (11) plaatsvinden. Als (10) gebeurt (Marie beslist of ze gaat gooien), dan brengt dit ons met kans 0.5 in een staat $\{Gooit(Jan), Gooit(Marie)\}$ en met kans 0.5 blijven we in staat $\{Gooit(Jan)\}$.
- Indien we in deze laatste staat terecht komen, kan hier enkel gebeurtenis (11) nog plaatsvinden. Met kans 0.6 resulteert dit in het breken van het raam, dit wil zeggen dat de toestand $\{Gooit(Jan), Gebroken\}$ bereikt wordt met kans 0.6, terwijl met kans 0.4 de toestand $\{Gooit(Jan)\}$ blijft. Aangezien er in deze beide toestanden geen gebeurtenissen meer kunnen plaatsvinden, zijn dit twee mogelijke eindtoestanden van het proces.

Door de andere mogelijkheden op een gelijkaardige manier uit te werken, bekomen we uiteindelijk het probabilistisch proces dat in Figuur 4 getoond wordt. We noemen een dergelijk proces een *uitvoeringsmodel* van de CP-theorie. Als we de mogelijke eindtoestanden beschouwen die dit proces kan bereiken, zien we dat hierover volgende kansverdeling bestaat:

$\{G(M), G(J), B\}$	$\{G(M), G(J)\}$	$\{G(J), B\}$	$\{G(J)\}$
0.46	0.04	0.3	0.2

Het proces uit Figuur 4 is niet de enige manier waarop de gebeurtenissen uit dit voorbeeld zouden kunnen plaatsvinden. Zo is het eveneens mogelijk dat, bijvoorbeeld, Marie haar steen voor Jan gooit of dat Jans steen het raam al raakt voor Marie geworpen heeft. Figuur 5 toont een ander uitvoeringsmodel voor ditzelfde voorbeeld. We zien hier dat, hoewel de interne structuur van dit proces sterk verschilt van die van het voorgaande, de uiteindelijke kansverdeling over zijn mogelijke eindtoestanden precies hetzelfde is als die in bovenstaande



Figuur 5: Een ander uitvoeringsmodel voor Voorbeeld 1.

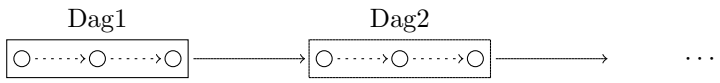
tabel. Het kan aangetoond worden dat dit een algemene eigenschap van onze taal is.

Stelling 1. *Als C een CP-theorie is, dan definieert elk uitvoeringsmodel van C dezelfde kansverdeling over mogelijke eindtoestanden.*

Deze stelling toont aan dat een CP-theorie niet alleen een klasse van probabilistische processen beschrijft, maar ook een unieke kansverdeling over eindtoestanden. Causale informatie is, met andere woorden, precies het soort informatie over een probabilistische proces dat je nodig hebt om de eindtoestand van dit proces te kunnen voorspellen. Dit is een interessant resultaat, dat bovendien een verband legt tussen onze dynamische aanpak van causaliteit en meer traditionele methodes zoals die van Pearl (2000), waarover we in Sectie 3.4 verder zullen uitweiden.

3.2 Tijd in CP-logica

Het soort van processen dat we tot nu toe beschouwd hebben, is nog vrij beperkt. Eén van de belangrijkste beperkingen is dat er nergens een notie van *tijd* aanwezig is. Als we bijvoorbeeld spraken over borstpijn, dan was dat borstpijn in het algemeen en niet, bijvoorbeeld, borstpijn op maandagmorgen om 9u. Er zijn tal van situaties waarin een dergelijke zienswijze niet voldoet. Denken we bijvoorbeeld maar aan het observeren van een patiënt die gedurende langere tijd in een ziekenhuis verblijft. We willen dan zeker onderscheid kunnen maken tussen zijn toestand op verschillende dagen. Dit kan gebeuren door te kijken naar processen die zijn onderverdeeld in verschillende *tijdsmomenten*. Zo kunnen we voor ons ziekenhuis-voorbeeld tijdsmomenten $Dag1, Dag2, \ldots$ beschouwen.



Figuur 6: Een globaal proces bestaat uit een opeenvolging van lokale processen.

Het soort van proces dat we trachten te modelleren, heeft nu een structuur zoals deze in Figuur 6 getoond wordt. Concreet betekent dit dat het proces begint op tijdstip $Dag1$. Binnen deze dag kunnen er bepaalde gebeurtenissen plaatsvinden—bijvoorbeeld, een patiënt met longontstekening kan borstpijn krijgen. Nadat alle gebeurtenissen van deze eerste dag zijn afgelopen, vindt er een overgang van $Dag1$ naar $Dag2$ plaats, waarbij de toestand op het einde van $Dag1$ natuurlijk bepaalt wat de toestand aan het begin van $Dag2$ zal zijn. Gedurende $Dag2$ kunnen er dan weer een aantal dingen gebeuren, waarvan het uiteindelijke resultaat dan weer propageert naar $Dag3$, enzovoort. We krijgen dus een structuur waarbij er een *globaal* proces is, dat in volgorde de verschillende momenten doorloopt en er bovendien op elk specifiek tijdstip een *lokaal* proces plaatsvindt—een proces dat zich volledig binnen dit ene tijdstip afspeelt. Deze lokale processen zijn dan in wezen identiek aan het soort van tijdsloze processen dat we tot hiertoe beschouwd hebben.

We merken ook op dat we in deze context het onderscheid kunnen maken tussen twee verschillende soorten van gebeurtenissen:

- Langs de ene kant zijn er gebeurtenissen die behoren tot een lokaal proces en die zich dus volledig binnen één bepaald moment afspelen (op Figuur 6 worden deze voorgesteld met stippellijnen);
- Langs de andere kant zijn er de gebeurtenissen van het globale proces, die zorgen voor een propagatie vanuit één tijdstip naar een volgend (op Figuur 6 worden deze voorgesteld met volle lijnen).

De vraag die ons nu natuurlijk interesseert, is hoe we ook deze processen zouden kunnen beschrijven. Het blijkt dat de taal die we nu hebben hiervoor reeds volstaat. Ter illustratie beschouwen we volgende voorbeeld.

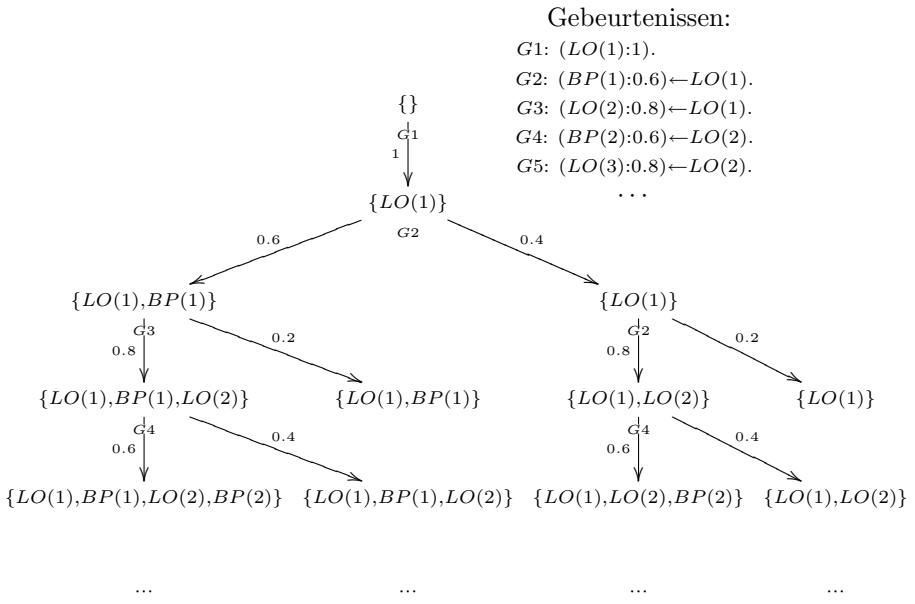
Voorbeeld 2. *Een patiënt komt het ziekenhuis binnen met een longontsteking. Het hebben van een longontsteking op een bepaalde dag veroorzaakt met kans* 0.6 *borstpijn op dezelfde dag. Bovendien zorgt een longontsteking op een bepaalde dag er met kans* 0.9 *voor dat de patiënt ook de volgende dag nog een longsteking zal hebben.*

$$Longsteking(1) \leftarrow . \tag{13}$$

$$\forall d\ (Borstpijn(d) : 0.6) \leftarrow Longsteking(d). \tag{14}$$

$$\forall d\ (Longsteking(d+1) : 0.9) \leftarrow Longsteking(d). \tag{15}$$

De notatie "$\forall d$" die in bovenstaande regels gebruikt wordt, is niets anders dan een korte manier om een heel aantal gelijkaardige CP-gebeurtenissen



Figuur 7: Initieel segment van het bedoelde uitvoeringsmodel van Voorbeeld 2.

op te schrijven. Zo stelt (14) bijvoorbeeld de volgende verzameling van CP-gebeurtenissen voor:

$$(Borstpijn(1) : 0.6) \leftarrow Longsteking(1).$$
$$(Borstpijn(2) : 0.6) \leftarrow Longsteking(2).$$
$$(Borstpijn(3) : 0.6) \leftarrow Longsteking(3).$$
$$\ldots$$

Regel (14) beschrijft nu een verzameling gebeurtenissen die zich alle volledig binnen hetzelfde tijdstip afspelen, terwijl (15) gebeurtenissen voorstelt die de toestand van een bepaald tijdstip propageren naar een volgend tijdstip. Gegeven onze bovenstaande discussie is het duidelijk dat we hier het proces trachten te beschrijven dat in Figuur 7 getoond wordt, dwz. het proces waarbij er eerst een lokaal proces voor dag 1 plaatsvindt (regels (13) & (14)), er dan een propagatie gebeurt van dag 1 naar dag 2 (regel (15)), er dan weer een lokaal proces voor dag 2 gebeurt, enzovoort. In algemeen kan aangetoond worden dat het proces dat op deze manier verloopt inderdaad een uitvoeringsmodel van bovenstaande CP-theorie zal zijn en dat de semantiek van CP-logica ons in dit geval dus inderdaad het correcte resultaat geeft.

We merken hierbij op dat er mogelijk ook uitvoeringsmodellen kunnen zijn, waarin de gebeurtenissen niet in de juiste volgorde plaatsvinden (zo heeft bovenstaand voorbeeld ook een uitvoeringsmodel waarin eerst alle progagaties van

longontsteking doorheen de verschillende dagen plaatsvinden en er pas daarna borstpijn op de eerste dag ontstaat); het bestaan van dergelijke "foutieve" processen is echter niet relevant, aangezien we uit Stelling 1 weten dat zij toch enkel maar dezelfde kansverdeling zullen genereren als het juiste proces.

3.3 Negatie in CP-logica

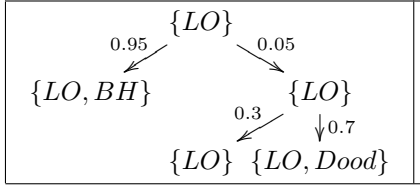
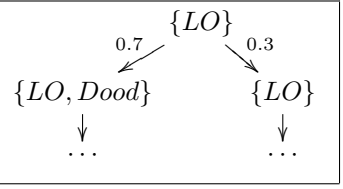
Tot slot breiden we de klasse van CP-theorieën die we beschouwen verder uit door nu ook het gebruik van negatie in de voorwaarden van gebeurtenissen toe te laten, dwz. dat nu ook de *af*wezigheid van een bepaalde eigenschap, het *niet* veroorzaakt worden hiervan, als oorzaak zal kunnen dienen. Om dit te illustreren beschouwen we volgend voorbeeld.

Voorbeeld 3. *Het hebben van een longontsteking heeft met kans* 0.95 *het effect dat de patiënt hiervoor behandeld wordt. Een niet behandelde longontsteking heeft met kans* 0.7 *de dood tot gevolg.*

$$(Behandeling : 0.95) \leftarrow Longontsteking. \tag{16}$$

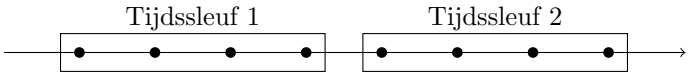
$$(Dood : 0.7) \leftarrow Longontsteking \wedge \neg Behandeling. \tag{17}$$

Als we ervan uitgaan dat de patiënt initiëel een longontsteking heeft, dan toont volgende figuur twee mogelijke uitvoeringsmodellen voor dit voorbeeld:

(16) – (17)	(17) – (16)
	

In het linker proces wordt er eerst beslist of de patiënt al dan niet behandeld wordt, waarna hij, in het geval van geen behandeling, overlijdt met kans 0.7. In het rechter proces, daarentegen, vindt de gebeurtenis waarmee de patiënt kan overlijden het eerst plaats. Dit is immers ook een valabele mogelijkheid, aangezien de patiënt in de begintoestand inderdaad een onbehandelde longontsteking heeft. Het effect hiervan is echter dat de patiënt met kans 0.7 overlijdt. Volgens dit tweede proces zijn de overlevingskansen dus veel slechter. Het is duidelijk dat het verschil in de structuur van deze twee processen nu dus wél een invloed uitoefent op de kans waarmee de mogelijke eindtoestanden bereikt worden. Als zodanig creëren we door het toelaten van negatie een ambiguïteit, die ervoor zorgt dat onze gewenste uniekheidseigenschap (Stelling 1) verloren gaat.

Betekent dit nu dat we niet anders kunnen dan negatie helemaal uitsluiten? Nee, het zal blijken te volstaan om enkel maar bepaalde beperkingen aan het gebruik van negatie op te leggen. De kern hiervan ligt in het onderscheid dat we reeds eerder gemaakt hebben tussen, langs de ene kant, gebeurtenissen die zich volledig binnen een bepaald tijdstip afspelen en, langs de andere kant, gebeurtenissen die van één tijdstip naar een volgend gaan. De ambiguïteit die



Figuur 8: Opdeling in tijdssleuven.

voortkomt uit het gebruik van negatie in het eerste soort van gebeurtenissen is onoplosbaar, dwz. dat er geen manier bestaat om te achterhalen welk het "juiste" antwoord zou moeten zijn. Bij de tweede soort van gebeurtenissen kan dit echter wel. Inderdaad, laat ons veronderstellen dat (17) een gebeurtenis van dit laatste soort is. Zo zou het bijvoorbeeld een gebeurtenis kunnen zijn die plaatsvindt gedurende de nacht van dag 1 op dag 2, dwz. dat de persoon die deze regel opschreef hiervoor volgende interpretatie in gedachten had:

> Als de patiënt op het einde van dag 1 een longontsteking heeft en hiervoor niet behandeld is, dan zorgt dit er met kans 0.7 voor dat hij tegen de volgende morgen overleden is.

In dit geval is het duidelijk welk van de twee bovenstaande processen het juiste is. We weten nu namelijk dat, ten eerste, de propositie *Behandeling* verwijst naar een eigenschap op dag 1 en, ten tweede, dat gebeurtenis (17) pas na deze dag kan plaatsvinden. Hieruit volgt dus dat het linker proces, dat waarin de patiënt eerst behandeld kan worden en dan pas kan overlijden, het juiste is.

Het is dan ook enkel in dit soort van gebeurtenissen—die waarin een propagatie van één tijdstip naar een later tijdstip gebeurt—dat we het gebruik van negatie zullen toelaten. We leggen dan aan onze uitvoeringsmodellen de bijkomende voorwaarde op dat een gebeurtenis van de vorm

$$(p_1 : \alpha_1) \vee \cdots \vee (p_n : \alpha_n) \leftarrow \varphi$$

enkel maar mag plaatsvinden, indien dat deel van het proces waardoor de waarheidswaarde van φ bepaald wordt reeds volledig is afgelopen. Het kan worden aangetoond dat het toevoegen van deze voorwaarde er voor zorgt dat onze uniekheidseigenschap (Stelling 1) herwonnen wordt. Bovendien kan het eveneens bewezen worden dat, indien het gebruik van negatie inderdaad beperkt blijft tot het juiste soort van gebeurtenissen, deze voorwaarde wel degelijk de juiste oplossing zal opleveren.

Tot slot van deze sectie beschouwen we nog een laatste voorbeeld.

Voorbeeld 4. *We beschouwen een tijdslijn die is onderverdeeld in een aantal verschillende tijdssleuven, zoals geïllustreerd in Figuur 8. Er zijn twee computers, een* cliënt *en een* server*. Gedurende de eerste tijdssleuf zal de cliënt met een bepaalde kans een aanvraag sturen naar de server. Als de server een aanvraag ontvangt, zal hij deze binnen dezelfde tijdssleuf aanvaarden of verwerpen; als hij ze aanvaard zal hij, nog steeds binnen dezelfde tijdssleuf, een antwoord terugsturen. Een boodschap die in een bepaalde tijdssleuf verstuurd wordt, zal*

ofwel in dezelfde tijdssleuf bij de ontvanger aankomen, ofwel pas in de volgende tijdssleuf aankomen, ofwel helemaal niet aankomen. Als de cliënt een aanvraag verstuurd heeft en hierop in dezelfde tijdssleuf geen antwoord ontvangt, zal hij in de volgende tijdssleuf zijn aanvraag herhalen. We modelleren dit voorbeeld met behulp van een alfabet, waarin het laatste argument van een predikaat telkens verwijst naar de tijdssleuf in kwestie.

$$(Zendt(Cl, Vraag, Srv, 1) : 0.7) \leftarrow . \tag{18}$$

$$\forall t\ (Aanvaardt(t) : 0.5) \vee (Verwerpt(t) :0.5) \leftarrow Krijgt(Srv, Vraag, t). \tag{19}$$

$$\forall t\ (Zendt(Srv, Antwoord, Cl, t) : 1) \leftarrow Aanvaardt(t). \tag{20}$$

$$\forall t, o, b, z\ (Krijgt(o, b, t) : 0.8) \vee (Krijgt(o, b, t + 1) : 0.1) \leftarrow Zendt(z, b, o, t). \tag{21}$$

$$\forall t\ Zendt(Cl, Vraag, Srv, t + 1) \leftarrow Zendt(Cl, Vraag, Srv, t) \wedge \neg Krijgt(Cl, Antwoord, t). \tag{22}$$

We merken hierbij op dat:

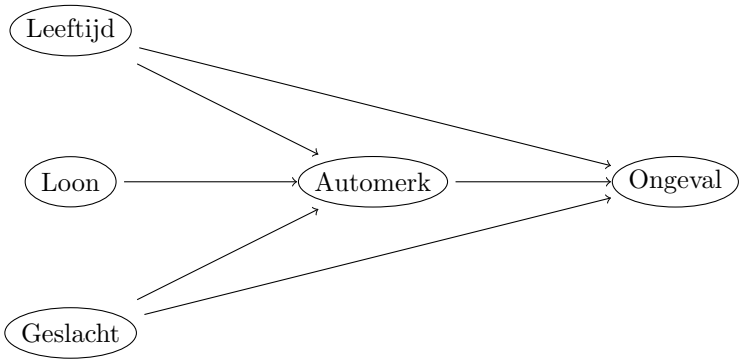
- (18), (19) en (20) gebeurtenissen zijn die zich volledig binnen dezelfde tijdssleuf afspelen;
- (21) een gebeurtenis is die zich zowel binnen één tijdssleuf als tussen twee tijdssleuven zou kunnen afspelen;
- (22) een gebeurtenis is die zich tussen twee tijdssleuven afspeelt.

Als zodanig is (22) hier (als enige) een gebeurtenis waarbinnen het gebruik van negatie inderdaad is toegestaan.

3.4 De relatie tot het werk van Pearl

In zijn invloedrijke analyse van causaliteit (Pearl 2000), maakt Pearl gebruikt van de formele taal van Bayesiaanse netwerken (en de veralgemening daarvan tot structurele modellen). Een dergelijk netwerk is een gerichte acyclische grafe, die een aantal probabilistisch (on-)afhankelijkheden uitdrukt. Concreet is elke knoop rechtstreeks afhankelijk van zijn ouders in de grafe, en enkel van deze. Het Bayesiaanse netwerk in Figuur 9 zegt bijvoorbeeld dat:

- Het merk van de auto waarmee een persoon rijdt een rechtstreekse invloed heeft op de kans dat hij in een auto-ongeval terecht komt;
- Leeftijd en geslacht van een persoon probabilistisch onafhankelijk zijn;
- Het loon van een persoon een *on*rechtstreekse invloed heeft op de kans waarmee deze een ongeval heeft, dwz. het hééft een invloed, maar enkel maar doordat het de keuze van automerk beïnvloedt. Moest de persoon bijvoorbeeld met een tijdelijke vervangwagen rijden, waarvan het merk



Figuur 9: Een Bayesiaans netwerk.

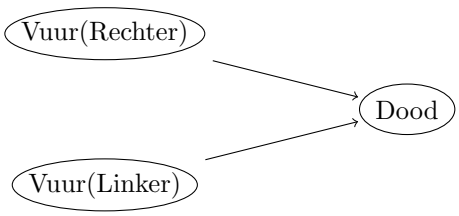
niet afhangt van zijn eigen loon, dan zou dit effect wegvallen en zouden het loon en de kans op een ongeval wél onafhankelijk zijn.

Gegeven deze onafhankelijkheden, kunnen we een nu unieke kansverdeling definiëren door aantal voorwaardelijke kansen vast te leggen. Concreet moeten we voor elke mogelijke waarde die een bepaalde knoop kan aannemen, de voorwaardelijke kans op deze waarde specifiëren, gegeven elke mogelijke toekenning van waardes aan zijn ouderknopen. Zo zouden we voor de knoop *Automerk* bijvoorbeeld volgende tabel kunnen opstellen:

	Merk=BMW	Merk=Kia	$\cdots$
Lftd=Jong,Gslcht=M,Loon=Hoog	0.6	0.1	
Lftd=Jong,Gslcht=M,Loon=Laag	0.2	0.3	
$\cdots$	$\cdots$	$\cdots$	

De basisfilosofie achter Bayesiaanse netwerken is duidelijk verschillend van die van CP-logica. Er is hier immers geen sprake van dynamische concepten als gebeurtenissen of processen, maar enkel van statische uitspraken over probabilistische afhankelijkheden. Toch kunnen we ook hier het dynamische aspect van causaliteit terugvinden. Het is immers mogelijk om een dergelijk netwerk te beschouwen als een (erg geabstraheerde) voorstelling van een proces, waarbij er voor elke knoop een gebeurtenis plaatsvindt, die de waarde van deze knoop bepaalt aan de hand van zijn ouders. Zo beschrijft het netwerk in Figuur 9 een proces waarbij eerst het geslacht, de leeftijd en het loon van een persoon bepaald worden, waarna de persoon dan, uitgaande van deze drie kenmerken, een auto van een bepaald merk uitkiest en zich hiermee in het verkeer begeeft, wat uiteindelijk met een bepaalde kans een ongeval tot gevolg heeft.

Het valt hierbij echter ogenblikkelijk op dat Bayesiaanse netwerken een nogal rigide structuur opleggen aan het soort van gebeurtenissen dat kan worden voorgesteld. Concreet zien we dat:



Figuur 10: Een Bayesiaans netwerk voor Voorbeeld 5.

- Er voor elke knoop precies één gebeurtenis moet zijn die de waarde van deze knoop volledig bepaalt;
- Alle gebeurtenissen moeten zorgen voor een propagatie in dezelfde richting.

Het volgende voorbeeld beschrijft een domein waarin de eerste van deze twee beperkingen niet geldt.

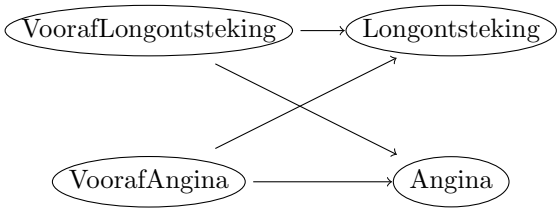
Voorbeeld 5. *Een speler speelt een spel Russische roulette met twee revolvers, een in zijn linkerhand en een in zijn rechter-. Beide revolvers hebben zes kamers, waarvan er precies één een kogel bevat. In CP-logica kan de relatie tussen het afvuren van deze twee revolvers en de dood van de speler als volgt worden voorgesteld:*

$$(Dood : 1/6) \leftarrow Vuur(Linker).$$
$$(Dood : 1/6) \leftarrow Vuur(Rechter).$$

Om dit voorbeeld als een Bayesiaans netwerk voor te stellen, moeten we deze twee afzonderlijke CP-gebeurtenissen te combineren tot één gebeurtenis, zoals getoond wordt in Figuur 10. De bijhorende voorwaardelijke kans tabel ziet er als volgt uit:

	Dood	¬Dood
Links, Rechts	11/36	25/36
¬ Links, Rechts	1/6	5/6
Links, ¬ Rechts	1/6	5/6
¬ Links, ¬ Rechts	0	1

Het voornaamste verschil tussen deze voorstellingswijze en die van CP-logica is dat de onafhankelijkheid tussen de twee mogelijke oorzaken voor de dood van de speler in CP-logica een *structurele* eigenschap was, terwijl ze in een Bayesiaans netwerk een *numerieke* eigenschap wordt (de twee oorzaken zijn onafhankelijk omdat $\frac{11}{36} = \frac{1}{6} + \frac{1}{6} - \frac{1}{6} \cdot \frac{1}{6}$). Dit maakt de voorstelling van CP-logica meer compact, aangezien ze slechts twee kanswaarden vereist ten opzichte van de vier in bovenstaande tabel. Bovendien is de CP-logica voorstelling ook meer *uitweidingstolerant*, dwz. dat het toevoegen (of verwijderen) van een oorzaak voor de dood van de speler op een eenvoudige manier kan gebeuren, door slechts het toevoegen (of verwijderen) van een nieuwe CP-gebeurtenis.



Figuur 11: Een Bayesiaans netwerk voor Voorbeeld 6.

Het volgende voorbeeld presenteert een geval waarin de tweede van de hierboven vermelde beperkingen geschonden wordt.

Voorbeeld 6. *Een longontsteking kan met kans* 0.2 *een oorzaak zijn voor* angina pectoralis. *Omgekeerd kan angina echter ook met kans* 0.1 *een longontsteking veroorzaken.*

$$(Angina : 0.2) \leftarrow Longontsteking.$$
$$(Longontsteking : 0.1) \leftarrow Angina.$$

In dit voorbeeld vinden we zowel een gebeurtenis terug die de waarde van *longontsteking* propageert naar *angina* als een die in de omgekeerde richting werkt. In CP-logica stelt dit geen problemen. De betekenis van bovenstaande CP-theorie is dan ook inderdaad wat men zou verwachten:

- Indien de patiënt noch angina noch een longontsteking al heeft opgelopen via een externe oorzaak, dan zullen beide gebeurtenissen niet plaatsvinden en heeft de patiënt dus geen van beide aandoeningen.
- Indien de patiënt al een longontsteking heeft, dan zal de eerste gebeurtenis plaatsvinden en krijgt hij met kans 0.2 ook angina.
- Indien de patiënt al angina heeft, dan zal de tweede gebeurtenis plaatsvinden en krijgt hij met kans 0.1 ook een longontsteking.
- Indien de patiënt zowel longontsteking als agina reeds heeft opgelopen, dan heeft hij natuurlijk beide.

Om hetzelfde resultaat te bekomen in een Bayesiaans network, moeten we het gewenste gedrag expliciet encoderen. Dit zou kunnen gebeuren door het introduceren van nieuwe eigenschappen $VoorafLongontsteking$ en $VoorafAngina$, waarmee dan het Bayesiaanse netwerk uit Figuur 11 gemaakt kan worden. De tabel die bij de knoop *Angina* hoort, ziet er als volgt uit (die voor *Longontsteking* is analoog):

	Angina	¬ Angina
VoorafAng, VoorafLong	1	0
VoorafAng,¬VoorafLong	1	0
¬VoorafAng, VoorafLong	0.2	0.8
¬VoorafAng,¬VoorafLong	0	1

Samenvattend kunnen we CP-logica dus zien als een verfijning van Bayesiaanse netwerken, die een meer flexibele voorstelling van causale gebeurtenissen biedt. Zoals we hebben aangetoond, levert dit een aantal voordelen op bij het het modelleren van domeinen waarvan de relevante gebeurtenissen niet onmiddellijk in de rigide structuur van een Bayesiaans netwerk passen.

3.5 Het verband met logische programma's

Er zijn een aantal voor de hand liggende gelijkenissen tussen zowel de syntax van CP-logica en die van logische programma's, als tussen de causale processen uit dit hoofdstuk en de constructieve processen uit het eerste deel van deze tekst. In deze sectie onderzoeken we dit verband nader. De kern van onze resultaten hierover ligt in een andere, equivalente manier om de semantiek van CP-logica te definiëren. Uit een gegeven CP-theorie kunnen we een logisch programma construeren door voor elke CP-gebeurtenis een onafhankelijke probabilistische keuze te maken. Om dit te illustreren, beschouwen we volgend voorbeeld:

$$(Herstel : 0.8) \vee (Dood : 0.1) \leftarrow Operatie. \tag{23}$$

$$(Operatie : 0.2) \leftarrow . \tag{24}$$

$$\tag{25}$$

Voor de eerste CP-gebeurtenis maken we nu volgende probabilistische keuze:

- Ofwel vervangen we haar door de regel $Herstel \leftarrow Operatie$ en dit doen we met kans 0.8;
- Ofwel vervangen we haar door de regel $Dood \leftarrow Operatie$ en dit doen we met kans 0.1;
- Ofwel laten we haar gewoon weg met de overgebleven kans van 0.1.

Daarna maken we voor de tweede gebeurtenis een gelijkaardige keuze. Zo construeren we uiteindelijk dan één van de volgende zes logische programma's:

Programma	Kans op programma
$Herstel \leftarrow Operatie.$ $Operatie \leftarrow .$	$0.8 \cdot 0.2$
$Operatie \leftarrow.$	$0.2 \cdot 0.2$
$Herstel \leftarrow Operatie.$	$0.8 \cdot 0.8$
...	...

De aldus geconstrueerde logische programma's interpreteren we nu volgens de gefundeerde model semantiek. Het volgende resultaat maakt dan een fundamentele verbinding tussen CP-logica en logisch programmeren.

Stelling 2. *De kans waarmee het hierboven beschreven proces uit een CP-theorie C een logisch programma construeert waarvan het gefundeerde model een bepaalde interpretatie I is, is precies gelijk aan de kans op deze interpretatie I volgens de CP-logica semantiek van de theorie C.*

Omdat CP-logica een taal is die we volledig hebben opgebouwd vanuit onze initiële analyse van causale uitspraken, toont deze stelling aan dat we ook uitdrukkingen uit logische programma's op een causale manier mogen interpreteren. Zo zien we bijvoorbeeld dat een regel uit een normaal logisch programma begrepen kan worden als een beschrijving van een deterministische causale gebeurtenis. Met andere woorden, we kunnen een regel $P \leftarrow B_1, \ldots, B_n$ uit een logisch programma lezen als:

> "Als alle eigenschappen B_1,...,B_n gelden, dan zal dit een gebeurtenis veroorzaken die P als haar effect heeft."

Deze observatie expliciteert een verband tussen logische programma's en causaliteit, dat reeds lang impliciet in dit domein aanwezig is; hierbij denken we bijvoorbeeld aan het gebruik van logische programma's voor het redeneren rond acties in de situatie calculus (Pinto en Reiter 1993). Bovendien legt dit resultaat ook een verband tussen CP-logica en talen uit het domein van probabilistische logisch programmeren. Eén van de implicaties van bovenstaande stelling is, bijvoorbeeld, dat de *onafhankelijke keuze logica* (*independent choice logic*) (Poole 1997) een deeltaal is van CP-logica, waaruit volgt dat we een theorie in deze logica kunnen begrijpen als een verzameling van onvoorwaardelijke probabilistische gebeurtenissen, gecombineerd met een verzameling deterministische causale gebeurtenissen.

3.6 Conclusies

De voornaamste bijdrage van ons werk rond CP-logica is dat we hiermee een taal ontwikkeld hebben, waarvan de syntax en semantiek op een natuurlijke manier volgt uit de eigenschappen van causale uitspraken. Concreet hebben we door onze analyse van dergelijke uitspraken het concept van een *causale probabilistische gebeurtenis-beschrijving* kunnen identificeren als een belangrijke component van causale kennis. Dit concept zelf suggereert reeds een semantiek in termen van probabilistische processen, die we dan ook formeel hebben uitgewerkt. Onze semantiek blijkt bovendien nauw gerelateerd te zijn aan logische programma's, wat ons in staat heeft gesteld om ook voor een aantal varianten van logische programmeren een causale interpretatie uit te werken.

Aangezien CP-logica reeds bepaalde voordelen biedt ten opzichte van Bayesiaanse netwerken en bovendien even expressief is als Pooles onafhankelijke keuze logica, zouden we verwachten dat deze taal ook effectief geschikt is voor

het modelleren van praktische toepassingen. Het onderzoeken hiervan zou bovendien, in het bijzonder, interessant zijn omdat we er zo achter kunnen komen hoe het soort van causale uitspraken dat we onderzocht hebben een rol speelt in concrete probleemdomeinen. Er zijn nog een aantal mogelijkheden voor toekomstig werk, die CP-logic beter geschikt zouden maken voor dergelijke praktische toepassingen.

Ten eerste kent CP-logica momenteel nog een aantal beperkingen, die het voorstellen van kennis nodeloos bemoeilijken. Zo kunnen er momenteel slechts een *eindig* aantal causale probabilistische gebeurtenissen beschouwd worden. In de praktijk botst men al snel op deze beperking. Laten we het eenvoudige voorbeeld beschouwen van een speler die net zolang een dobbelsteen rolt tot hij zes werpt. Er bestaat dan geen eindige bovengrens op het aantal worpen en daarom kan dit voorbeeld momenteel niet gemodelleerd worden in CP-logica. Het is evenmin mogelijk om gebeurtenissen voor te stellen die enkel een *bijdragend* effect hebben. Bijvoorbeeld het open draaien van een kraan zal er niet voor zorgen dat een badkuip ogenblikkelijk vol is, maar zal enkel een bepaalde hoeveelheid water per seconde bijdragen. Tot slot kunnen er ook enkel maar gebeurtenissen worden voorgesteld, waarvan de verzameling van mogelijke uitkomsten al op voorhand volledig vastligt. Ook dit is in de praktijk vaak niet zo.

Ten tweede is er, naast het verbeteren van CP-logica zelf, ook het onderwerp van het integreren hiervan in een groter kennisrepresentatie formalisme. Voor praktische doeleinden lijkt dit haast onontbeerlijk. Het opstellen van een correcte theorie in CP-logica vereist immers dat alle relevante gebeurtenissen, oorzaken en gevolgen precies gekend zijn. Het is onrealistisch om te veronderstellen dat dit in een effectieve toepassing voor het hele probleemdomein zo zal zijn. In plaats daarvan is er typisch slechts hier en daar een deelaspect waarover dergelijke kennis aanwezig is. Om CP-logica ook in dergelijke omstandigheden te kunnen gebruiken, moet deze taal dus gecombineerd worden met andere vormen van kennis. Er zijn een aantal voor de hand liggende kandidaten hiervoor, zoals: uitspraken over de kans op bepaalde eigenschappen, uitspraken over probabilistische onafhankelijkheden of beperkingen op de mogelijke toestanden waarin de wereld zich kan bevinden. Het integreren van dergelijke verschillende vormen van kennis—zonder conceptuele duidelijkheid te verliezen—is dan ook één van de grootste uitdagingen voor ons werk rond CP-logica, en misschien zelfs voor het vakgebied van onzekerheid in artificiële intelligentie in het algemeen.

Ten derde is er ook verdere studie rond inferentie methodes wenselijk. De meest voor de hand liggende inferentie-taak voor CP-logica is het deductief afleiden van de kans op bepaalde formules. Dit kan op een eenvoudige manier gebeuren, door ons te baseren op het verband tussen CP-logica en logische programma's, en dergelijke kansen te berekenen door een gepast gebruik van gekende algoritmes uit logisch programmeren. Het is echter zinvol om ook snellere methodes dan deze naïeve aanpak uit te werken. Een tweede inferentie-taak betreft het opstellen van een theorie. Zoals bij probabilistische modellen bijna steeds het geval is, is het ook voor CP-logica niet wenselijk dat de gebruiker zelf allerhande kansen moet uitrekenen of schatten. In plaats daarvan dient het

mogelijk te zijn deze automatisch uit een databank af te leiden. Voor CP-logica zijn er reeds een aantal algoritmes ontwikkeld die hiertoe in staat zijn (Riguzzi 2004; Blockeel en Meert 2007), als de theorie tenminste aan bepaalde beperkingen voldoet. Het zou interessant zijn om deze verder uit te breiden, zodat ze ook in het algemene geval kunnen worden toegepast. Ook het leren van de *structuur* van een theorie in CP-logica is een interessant onderwerp. Door na te gaan welke theorie het best een gegeven databank beschrijft, achterhalen we immers welke causale mechanismen er het sterkst in de data aanwezig lijken te zijn. Dit soort informatie kan voor praktische toepassing zeer relevant zijn. Wanneer men bijvoorbeeld in de bio-informatica werkzame stoffen tracht te isoleren, is dit precies het soort informatie dat nodig is.

4 Conclusies

In deze thesis hebben we aangetoond dat constructieve processen een belangrijke rol spelen in kennisrepresentatie. Langs de ene kant hebben we dergelijke processen gebruikt om op een hoog niveau de eigenschappen van bepaalde talen te analyseren. Concreet hebben we dit gedaan voor de onderwerpen van *modulariteit* en *predikaat introductie*, waar we telkens in staat waren om door middel van één algebraïsche stelling over dergelijke processen een aantal verschillende resultaten voor verschillende logica's, die in de literatuur allemaal afzonderlijk bewezen waren, te veralgemenen. Soms waren die veralgemeningen eerder voor de hand liggend (zoals het uitbreiden van een bestaand resultaat naar een grotere klasse van theorieën of een andere semantiek), maar in het geval van predikaat introductie voor logische programma's was de veralgemening van die aard, dat ze de interessante mogelijkheid schiep om universele kwantoren te vervangen door een recursieve lus. Samen tonen deze resultaten aan dat het bestuderen van constructieve processen binnen het algebraïsche kader van approximatie theorie een waardevolle methode is om eigenschappen van verschillende talen op een uniforme manier te analyseren.

Daarnaast hebben we ook aangetoond dat constructieve processen een belangrijke rol spelen bij een juist begrip van causale uitspraken. Causaliteit heeft immers een inherent dynamisch aspect, dat op een natuurlijke manier aanleiding geeft tot een formalizatie op basis van probabilistische processen, zoals we dit met CP-logica gedaan hebben. Door zijn flexibele voorstellingswijze van causale gebeurtenissen biedt deze taal een aantal voordelen ten opzichte van Bayesiaanse netwerken. Bovendien vertoont de semantiek van deze taal, zoals we ze uit onze analyse van causale uitspraken hebben afgeleid, een aantal bijzondere gelijkenissen met de gefundeerde model semantiek van logische programma, wat aantoont dat we ook uitdrukkingen uit deze taal, en bijvoorbeeld eveneens die uit Pooles onafhankelijke keuze logica, kunnen begrijpen als uitspraken over causale gebeurtenissen.

Referenties

Aravindan, C. en P. M. Dung (1995). On the correctness of unfold/fold transformation of normal and extended logic programs. *Journal of Logic Programming 24*(3), 201–217.

Blockeel, H. en W. Meert (2007). Two novel methods for learning logic programs with annotated disjunctions. In *Proceedings of the 16th International Conference on Inductive Logic Programming, ILP'06*, Lecture Notes in Computer Science. Te verschijnen.

Denecker, M., V. Marek, en M. Truszczyński (2003, January). Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence 143*(1), 79–122.

Denecker, M. en E. Ternovska (2004). A logic of non-monotone inductive definitions and its modularity properties. In V. Lifschitz en I. Niemelä (Eds.), *Seventh International Conference on Logic Programming en Nonmonotonic Reasoning, LPNMR'04*.

Dix, J. en M. Müller (1994). Partial evaluation and relevance for approximations of stable semantics. In Z. W. Ras en M. Zemankova (Eds.), *ISMIS*, Volume 869 of *Lecture Notes in Computer Science*, pp. 511–520. Springer.

Eiter, T., G. Gottlob, en H. Mannila (1997). Disjunctive datalog. *ACM Transactions on Database Systems (TODS) 22*, 364–418.

Gelfond, M. en H. Przymusinska (1992). On consistency and completeness of autoepistemic theories. *Fundamenta Informaticae 16*(1), 59–92.

Lifschitz, V. en H. Turner (1994). Splitting a logic program. In P. V. Hentenryck (Ed.), *International Conference on Logic Programming (ICLP'94)*, pp. 23–37. MIT Press.

Lloyd, J. en R. Topor (1984). Making Prolog more expressive. *Journal of Logic Programming 1*(3), 225–240.

Marek, V. W. en M. Truszczyński (1991). Autoepistemic logic. *Journal of the ACM 38*(3), 588–619.

Niemelä, I. en J. Rintanen (1994). On the impact of stratification on the complexity of nonmonotonic reasoning. *Journal of Applied Non-Classical Logics 4*(2).

Pearl, J. (2000). *Causality: Models, Reasoning, en Inference*. Cambridge University Press.

Pinto, J. en R. Reiter (1993). Temporal reasoning in logic programming: A case for the situation calculus. In *Proc. of the International Conference on Logic Programming*, pp. 203–221.

Poole, D. (1997). The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence 94*(1-2), 7–56.

Riguzzi, F. (2004, September). Learning logic programs with annotated disjunctions. In A. Srinivasan en R. King (Eds.), *14th Internation Conference on Inductive Logic Programming (ILP2004), Porto, 6-8 September 2004*, Heidelberg, Germany, pp. 270–287. Springer Verlag.

Turner, H. (1996). Splitting a default theory. In *Proc. Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pp. 645–651. AAAI Press.

Van Gelder, A. (1993). The alternating fixpoint of logic programs with negation. *Journal of Computer en System Sciences 47*(1), 185–221.

Verbaeten, S., M. Denecker, en D. De Schreye (2000). Compositionality of normal open logic programs. *Journal of Logic Programming 41*, 151–183.