

The ISP Course Selection Puzzle

Herbert Gorissen

Thesis voorgedragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
elektrotechniek, optie Elektronica en
geïntegreerde schakelingen

Promotor:

Prof. Gerda Janssens

Begeleider:

Matthias van der Hallen

© Copyright KU Leuven

Without written permission of the thesis supervisor and the author it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to ESAT, Kasteelpark Arenberg 10 postbus 2440, B-3001 Heverlee, +32-16-321130 or by email info@esat.kuleuven.be.

A written permission of the thesis supervisor is also required to use the methods, products, schematics and programs described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

Zonder voorafgaande schriftelijke toestemming van zowel de promotor als de auteur is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot ESAT, Kasteelpark Arenberg 10 postbus 2440, B-3001 Heverlee, +32-16-321130 of via e-mail info@esat.kuleuven.be.

Voorafgaande schriftelijke toestemming van de promotor is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Preface

Herbert Gorissen

Contents

Preface	i
Abstract	iii
List of Figures and Tables	iv
List of Abbreviations and Symbols	v
1 Introduction	1
1.1 Probleemstelling	2
1.2 Doel	3
1.3 Dependencies	5
2 Implementation	7
2.1 Constructie ISP Theory	7
2.2 Front-End	10
2.3 Features	11
2.4 Conflict Explanation	15
3 Evaluation	21
4 Conclusion	27
5 Related Work	29
6 Future Work	33
Bibliography	35

Abstract

Bij de start van een opleiding aan de KU Leuven is iedere toekomstige student verplicht zijn of haar opleiding samen te stellen uit een hele waaier aan opleidingsonderdelen. De selectie van opleidingsonderdelen is echter onderworpen aan een set van regels die allemaal voldaan moeten zijn wil men een geldig individueel studieprogramma (ISP) bekomen. Het ISP maakt deel uit van een specifieke groep van problemen genaamd configuratie problemen.

Het IDP systeem ontwikkeld aan de KU Leuven laat toe domein specifieke kennis uit te drukken in $FO(\cdot)$, een uitbreiding op eerste orde logica, ook wel de theorie genoemd. Eens de kennis beschreven is kan men met het bijhorende IDP systeem verscheidene vormen van inferentie toepassen op de theorie.

IDP als kennis representatie systeem is ideaal voor het beschrijven van en later redeneren over configuratie problemen. In het verleden heeft men er al grote configuratie problemen uit de bedrijfswereld succesvol mee kunnen beschrijven en oplossen. Interessant is om te achterhalen of we dit voor het ISP selectie probleem ook kunnen doen.

Momenteel zijn er bij het opstellen van het ISP ook een aantal tekortkomingen die we met IDP zouden willen oplossen. Zo wordt het lessenrooster niet mee in rekening gebracht, waardoor studenten niet weten of ze opleidingsonderdelen opnemen waarvan de lessen mogelijk kunnen overlappen. En wat als er een foutieve selectie wordt gemaakt? Momenteel krijgt de student na het bevestigen van diens selectie een foutmelding en deze is vaak nog onduidelijk. IDP voorziet momenteel een aantal functies om de oorzaken van inconsistentie op te sporen at run-time. Handig zou zijn om hiermee korter op de bal te spelen en meteen bij een foutieve selectie uitleg te kunnen geven waarom de voorlopige huidige selectie niet correct is en hoe men dit kan oplossen.

Naast dit alles heeft deze thesis nog een andere grote doelstelling. Het opsporen van oorzaken van inconsistenties ofwel conflict explanation is een gebied waar momenteel nog veel onderzoek gaande is. Enkele interessante technieken zijn hier al voorgelegd en het is in onze interesse om na te gaan of en hoe we deze kunnen integreren samen met het IDP systeem.

List of Figures and Tables

List of Figures

List of Tables

3.1	Functionaliteiten	22
-----	-------------------	----

List of Abbreviations and Symbols

Abbreviations

LoG	Laplacian-of-Gaussian
MSE	Mean Square error
PSNR	Peak Signal-to-Noise ratio

Symbols

42	“The Answer to the Ultimate Question of Life, the Universe, and Everything” according to
c	Speed of light
E	Energy
m	Mass
π	The number pi

Chapter 1

Introduction

1.0.1 Constraint Satisfaction Problems

CSP's zijn problemen waar voor een reeks variabelen een geldige waarde uit het domein dient toegekend te worden. De toekenning echter is gebonden aan een set van regels waaraan de variabelen moeten voldoen. Formeel definiëren we een CSP als een triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, met \mathcal{X} de verzameling van variabelen, \mathcal{D} de domeinwaarden voor deze variabelen en \mathcal{C} de constraints over de variabelen. Typisch worden dit soort problemen opgelost d.m.v. zoeken door alle mogelijk combinaties, deze recursieve methode gaat voor elke nieuwe toekenning na of de constraints voor de partiële toekenning consistent zijn. Zo ja, dan volgt er een nieuwe recursieve oproep, anders gaat het algoritme backtracken. Het oplossen van CSP's met een eindig domein behoort tot de klasse van NP-Complete problemen en de berekeningen zijn vaak van hoge complexiteit met betrekking tot de grootte van het domein. Om het zoekproces proberen te versnellen bestaan er variaties op backtracking zoals backmarking en backjumping. De eerste zorgt voor een efficiëntere manier om consistentie na te gaan en backjumping is een betere manier van backtracking waarbij men over meerdere waarden tegelijk kan backtracken. Deze technieken versnellen het zoekproces in dat ze onnodige stappen detecteren en overslaan. Anderzijds bestaan er propagatie technieken die lokale consistentie garanderen met technieken zoals arc consistency, hyper-arc consistency en path-consistency die de domeinen van de variabelen proberen te verkleinen alvorens het zoekproces gestart wordt.

1.0.2 Interactive Configuration Problems

In deze thesis ligt de focus op een specifieke groep van problemen binnen het domein van CSP's genaamd Interactieve Configuratie (IC) problemen. Constraint Satisfaction Problems zijn problemen waar een toekenning van domeinwaarden voor de variabelen dient gevonden te worden waarvoor geldt dat aan alle constraints voldaan is. IC problemen hebben hetzelfde doel, maar bij CSP's wordt er naar een toekenning gezocht d.m.v. recursieve zoektechnieken terwijl het in het geval van IC problemen een gebruiker is die handmatig stap per stap een geldige toekenning gaat proberen samenstellen. Om de gebruiker hierin bij te staan is er nood aan software, software

die de gebruiker zo goed mogelijk bijstaat gedurende dit hele proces. Uit ervaring is gebleken dat het schrijven van software voor dit soort problemen geen gemakkelijke opgave is. Zo is aangetoond dat een imperatieve aanpak voor het beschrijven van de regels van een probleem vaak moeilijk is (Gelle and Weigel, 1996). Reden hiervoor is dat de regels over de betreffende domeinkennis vaak verspreid zit in de software in codesnippets. Daarbij is onderhoud van zulke software een enorm moeilijke opgave waarbij de kleinste wijziging in de constraints kan leiden tot een volledige herwerking van de code. Een alternatief hiervoor is een declaratieve aanpak. In declaratieve toepassingen kunnen de regels van een probleem beschreven worden d.m.v. logische constraints. Gelle en Weigel Gelle and Weigel (1996) laten niet alleen zien dat de declaratieve beschrijving van een probleem overzichtelijker en duidelijker is maar ook dat een wijziging in de constraints gemakkelijk aangepast kan worden. En hoewel er met een imperatieve aanpak optimalere algoritmen ontwikkeld kunnen worden, is aangetoond dat declaratieve methoden ook goede prestaties kunnen neerzetten (?).

1.0.3 Knowledge Representation: Het IDP Systeem

Er zijn ondertussen vele declaratieve systemen verschenen (Prolog, Ant, Lisp, ...). Wat opvalt is dat elk systeem vaak gebonden is aan één enkele specifieke vorm van inferentie. Dus afhankelijk van de gewenste inferentie zal een probleem opnieuw moeten beschreven worden in een ander systeem waarmee deze vorm van inferentie wel mogelijk is, ondanks het feit dat de kennis telkens dezelfde is. Deze gedachtegang ligt aan de basis van het concept van Knowledge Representation (Denecker and Vennekens, 2008). In het KR-paradigma wordt gesteld dat de inferentie los staat van de kennis over een probleem. Deze kennis is niets meer dan een verzameling van informatie, maar met deze informatie kunnen meerdere vormen van inferentie toegepast worden. In dit onderzoek ligt de aandacht op IDP, een KR-systeem ontwikkeld aan de K.U. Leuven en voor het eerst voorgesteld in 2008. Het laat toe om de kennis over een probleem te beschrijven in $FO(\cdot)$, dit is eerste orde logica maar uitgebreid met aggregaten, type definities en inductieve definities. De grote uitdrukingskracht van $FO(\cdot)$ gecombineerd met de scheiding van kennis en inferentie zijn al meermaals voorgesteld als de grote pluspunten van het systeem ? ? ? ?.

1.1 Probleemstelling

1.1.1 Individueel Studieprogramma

Een ISP samentstellen valt onder de categorie van Interactieve Configuratieproblemen. De gebruiker, in dit geval een toekomstige student wil een geldig ISP bekomen d.m.v. het selecteren van mogelijke opleidingsonderdelen (vakken) voor de door hem/haar gekozen opleiding. Maar de student kan niet zomaar elk opleidingsonderdeel selecteren, er zijn een heleboel regels waaraan de selectie moet voldoen. Deze regels zijn op zichzelf duidelijk en intuïtief, maar een selectie vinden die aan alle regels voldoet kan mogelijk verwarrend zijn. Het huidige systeem, een web applicatie op de website van de universiteit voorziet wel enige ondersteuning gedurende het

selectieproces. De student kan simpelweg vakken selecteren door in vinkje te zetten in de checkbox dat bij een bepaald vak hoort, deze selectie kan ongedaan gemaakt worden door dit vinkje weer weg te klikken. Wanneer de student tevreden is met de selectie, kan hij/zij diens keuze bevestigen en het systeem zal dan controleren of de selectie voldoet aan de regels van het ISP. Is dit niet het geval dan wordt de student hiervan op de hoogte gebracht en wordt er uitleg (verklaring in natuurlijke taal) voorzien waarom ze niet klopt. Dit niveau van ondersteuning is op zich voldoende en iedere student zal uiteindelijk in staat zijn een geldig ISP samen te stellen. Maar de vraag is of we met IDP beter kunnen doen.

1.1.2 Conflict Explanation

In een interactief configuratieprobleem is een gebruiker verantwoordelijk voor de toekenning van de waarden aan de variabelen, en het is goed mogelijk dat deze keuze niet voldoet aan de regels van het probleem. Sinds het de verantwoordelijkheid is van het systeem om de gebruiker ondersteuning te bieden, moet het deze conflicten kunnen detecteren en aan de gebruiker kunnen duidelijk maken wat er precies scheelt zodanig dat ze opgelost kunnen worden. Hier komt nog eens bij dat IC problemen een snelle reactietijd vereisen, de gebruiker mag niet meer dan een paar seconden wachten op een antwoord. Meerdere technieken voor het verklaren en oplossen van conflicten zijn reeds voorgesteld ? ? (?). IDP zelf beschikt momenteel ook over enkele technieken voor conflict explanation, zo zijn er de unsatstructure en de unsattheory.

1.2 Doel

In deze thesis wil ik onderzoeken of de regels van het ISP efficiënt kunnen beschreven worden in $FO(\cdot)$. De regels zijn voor elke opleiding anders. De vraag is of het mogelijk is om met één enkele theorie in IDP meerdere opleidingen correct te kunnen beschrijven. Belangrijk hierbij is dat de domeinen evenals de regels sterk kunnen verschillen tussen opleidingen, en dat de eventuele theorie hier mee om moet kunnen gaan.

De K.U. Leuven heeft haar eigen systeem voor het samenstellen van het ISP. Een minpunt van dit systeem is de ondersteuning die het biedt tijdens het selectieproces. Hoewel ze voldoende is om de student bij te staan, ben ik ervan overtuigd dat een nieuwe applicatie op basis van IDP betere ondersteuning kan geven en meerdere mogelijkheden kan bieden. In een zelf ontworpen Front-end met grafische user interface wil ik de volgende functionaliteiten integreren:

Automatisch invullen van gevolgen Als de student een vak A kiest en hieruit volgt dat vak B ook gevolgt moet worden, dan is het de bedoeling dat het systeem dit automatisch invult zodat de student zich hier verder niets van hoeft aan te trekken.

Detectie van foutieve selectie Deze functionaliteit is momenteel al aanwezig in het huidige systeem. Maar de detectie gebeurt pas bij de bevestiging van de

selectie i.p.v. op het moment van de selectie zelf. En dat is wat ik zal proberen te integreren in de nieuwe front-end. En niet enkel dit, maar ook het effectief opsporen van de oorzaak zodat de gebruiker dit kan aanpassen.

Geldig ISP laten genereren Stel dat een student keuzes heeft gemaakt omtrend de vakken die hij/zij echt wil of niet wil volgen, maar de selectie is nog geen volledige oplossing. Dan kan de student vragen aan het systeem om de selectie verder in te vullen.

Optimaal ISP laten genereren Niet alleen moet het systeem een geldige oplossing kunnen genereren, maar ook de beste oplossing volgens een bepaalde criterium. Zo zou een student bijvoorbeeld graag een ISP willen waarbij de werklust zo goed mogelijk verdeeld is over beide semester.

Ongedaan maken van acties Deze functionaliteit is terug te vinden in zowat de meeste moderne systemen. Als een student ontevreden is over zijn/haar recente keuzes, moet er de mogelijkheid zijn om deze ongedaan te kunnen maken.

Weergave van het lessenrooster Voorheen had ik al vermeld dat studenten geen duidelijk overzicht hebben van het lessenrooster dat voortvloeit uit hun keuzes. Vakken kunnen lesmomenten hebben die mogelijk overlappen met die van andere vakken en dit kan voor een ongewenste verrassing zorgen eens het ISP bevestigd is. Het is mijn bedoeling om in de nieuwe front-end wel een eventueel lessenrooster weer te geven.

De meeste van deze functionaliteiten steunen op inferentie technieken die IDP aanbiedt zoals model expansie, minimalisatie, propagatie etc. Dit wil zeggen dat om deze functionaliteiten te kunnen aanbieden, de onderliggende inferentie technieken efficiënt moeten werken. Herinner dat we te maken hebben met een IC probleem en dit vereist dat de reactietijd niet meer dan enkele seconden bedraagt. Ik zal dus moeten onderzoeken of inferentie goed verloopt voor dit probleem.

Naast het testen van de mogelijkheden van IDP voor het ISP probleem, heeft deze thesis nog een andere doel. De notie van conflict explanation werd reeds aangehaald, en het is mijn doel om de doeltreffendheid van sommige methodes te testen op het ISP probleem. Het opsporen van foutieve selecties met de unsatstructure is één ding, maar kunnen *verklaren* wat er mis is met de selectie of oplossingen bieden is een ander verhaal. Verscheidene technieken zijn al onderzocht en voorgesteld en IDP zelf voorziet ook al de unsattheory, een methode die op zoek gaat naar de regels die (in geval van niet-satisfieerbaarheid) nooit waar gemaakt kunnen worden. Probleem is echter dat de output hiervan geformuleerd is in $FO(\cdot)$, en een doorsnee gebruiker zal niet in staat zijn om hieruit iets te kunnen afleiden. Reified constraints zouden hier een oplossing voor kunnen zijn. Dus de unsatstructure kan de oorzaak van het conflict aanwijzen, terwijl reified constraints als ze voldoen aan de verwachtingen een verklaring in natuurlijke taal kunnen geven voor de regels die overtreden zijn. Een laatste manier die kan helpen bij het oplossen van conflicten is het opsommen van mogelijke oplossingen. Met oplossing wordt bedoelt, de kleinste verzameling van

selecties die dienen ongedaan gemaakt te worden zodanig dat de rest van de selectie terug satisfieerbaar is. Om dit te kunnen implementeren baseer ik mij op het werk van (?).

1.3 Dependencies

FO(·) Het is de bedoeling om de mogelijkheden van IDP in actie te zien en met een proof of concept de prestaties van het systeem te testen. FO(·) bezit een grote uitdrukkringskracht, bovenop eerste order logica bevat het ook taalelementen zoals aggregaten (sum, avg, min, max, ..) en inductieve definities.

Kivy Bij de keuze van de programmeertaal en framework viel mijn oog op Kivy. Dit is een Python library met een hele waaier aan grafische elementen die elk uitgebreid geconfigureert kunnen worden. Hiervoor kan gebruik gemaakt worden van een door de uitgever ontwikkelde kv-language. Kivy applicaties werken cross-platform en de applicaties zijn event-driven gebruik makend van een centrale loop.

JSON Het gegevensformaat van JSON is ideaal om de domeinen van verschillende opleidingen in weer te geven. De domeinwaarden zitten gekoppeld aan een attribuut, dit maakt het gemakkelijk om structuur te parsen en zo het domein voor iederen opleiding in te lezen.

Java De eindige toestandsautomaat uit de paper van (?) is gebouwd in java, hierbij maak ik gebruik van de 'brics' automaton library. Deze library is simpel in gebruik en biedt handige tools die essentieel zijn voor de bouw van de automaat zoals FSA minimizatie.

Chapter 2

Implementation

2.0.1 Preliminaries

De kennis over het ISP staat beschreven in een theorie T over een vocabularium Σ . Dit vocabularium bestaat uit een set Σ_p van predicaatsymbolen en een set van functiesymbolen Σ_f . Een interpretatie I bestaat uit D de verzameling domeinelementen, een mapping van elk functiesymbool f/n naar een functie met ariteit n op D en een mapping van elk predicaatsymbool P/n naar een relatie $R \subseteq D^n$. Een three-valued partiële interpretatie bestaat uit een mapping op drie waarheidswaarden u, t, f respectievelijk onbekend, waar en niet waar. De waarheidswaarden zijn partieel geordend volgens precisie, $u \leq_p f$ en $u \leq_p t$. De bedoeling is om een exacte interpretatie te bekomen met maximale precisie die consistent is met de theorie T .

2.1 Constructie ISP Theory

Analyse van het ISP Alvorens te beginnen aan het opstellen van een theorie in IDP, is het belangrijk om de regels en structuren van het ISP te bestuderen. Het doel is om een theorie te ontwikkelen die zo algemeen mogelijk is, zodanig dat ze in staat is om een correcte interpretatie te beschrijven voor verschillende opleidingen. En het vinden van structuren, patronen, regels en types die vaak (of altijd) terugkomen, kan in grote mate bijdragen aan zo'n goede theorie. Omdat het onderwijsaanbod aan de K.U. Leuven zodanig groot is, heb ik mijn analyse beperkt tot een kleine groep opleidingen binnen de faculteit Wetenschappen en Ingenieurswetenschappen. Wat meteen opvalt is dat bepaalde types van elementen telkens voorkomen namelijk vakken (meer specifiek de vakcode), studiepunten, fases en semesters. Vakken maken altijd deel uit van een groep, en voor verschillende groepen gelden vaak andere regels. Zulke groepen of anders genoemd VakGroepen behoren tot verschillende VakgroepTypes (elke groep behoort tot één uniek type) en afhankelijk van het type kunnen er mogelijk nog extra constraints gelden voor een VakGroep. Deze types die steeds terugkomen in meerdere opleidingen hebben ongeacht de opleiding waar ze in voorkomen dezelfde regels die ermee gepaard gaan. Zo is elke opleiding op zich een vakgroep van het type opleiding, het bevat vakken die je verplicht bent te volgen en keuzevakken. Het heeft een minimum (en

maximum) aantal studiepunten dat de student verplicht moet opnemen. Een ander belangrijk type vakgroep is de (hoofd)specialisatie, verscheidene opleidingen geven de keuze tussen meerdere specialisaties. Verwacht wordt dat je van minstens één zo'n specialisatie alle verplichte vakken opneemt. Naast hoofdspecialisatie bestaat er ook het type verdere specialisatie, waarin de student verplicht wordt een bepaald aantal studiepunten op te nemen aan vakken uit deze of bepaalde andere vakgroepen. Vervolgens is er het type 'Algemeen vormende en onderzoeksondersteunende groep' vakgroep, dit is veruit het moeilijkste type groep om te beschrijven. De student moet opnieuw een bepaald aantal studiepunten aan vakken opnemen uit deze groep. Maar daarnaast gelden er ook specifieke regels voor verscheidene vakken die deel uitmaken ervan. En als laatste is er het type 'Bachelor verbredend pakket', waarin studenten die beginnen aan hun masteropleiding vakken moeten opnemen die ontbraken in hun bacheloropleiding. Deze types zien we het vaakst voorkomen en kunnen ongeacht de opleiding met dezelfde set van regels beschreven worden. Het is mogelijk dat sommige types van vakgroepen niet telkens voorkomen, en dus de regels ook niet van kracht zijn. In een bacheloropleiding zal bijvoorbeeld nooit een vakgroep voorkomen van het type Bachelor verbredend pakket.

Constructie van de theorie De resultaten van de analyse moeten vervolgens verwerkt worden in een vocabularium dat gebruik maakt van deze types en structuren. Op basis van dat vocabularium hoort dan een theorie ontwikkeld te worden die in staat is een geldige interpretatie voor de verschillende opleidingen te beschrijven.

2.1.1 Vocabularium

Om te beginnen zal ik de structuur dit vocabularium toelichten, hieronder vallen de types, predicaten en functies.

Types

Vak Dit type omvat de verzameling van de verschillende vakken in het domein, elke opleiding bestaat uit verschillende vakken die een gebruiker mogelijk kan selecteren. Meer specifiek zijn dit de unieke vakcodes van een vak. De naam van een vak is namelijk niet altijd uniek in tegenstelling tot de vakcode. De domeinwaarden zijn niet de naam in natuurlijke taal, maar de unieke vakcode.

VakGroep Vakken maken altijd deel uit van een groep, en per VakGroep gelden vaak unieke regels.

VakGroepType Elke vakGroep maakt deel uit van een bepaald type groep, en per type groep gelden er bepaalde regels bovenop de regels van de individuele groep zelf.

Fase Opleidingen bestaan uit fases oftewel jaren, de Master Computer Wetenschappen bestaat bijvoorbeeld uit 2 fases. Vakken binnen een opleiding kunnen enkel gevolgd worden tijdens de fase van de opleiding waartoe ze behoren.

Semester De K.U. Leuven werkt met een semester systeem, dit wil zeggen dat elke fase (schooljaar) is opgedeeld in 2 semesters. Elk vak kan gevolgd worden in de semester waartoe het behoort. Er bestaan ook jaarvakken waarvoor de werklast verdeeld is tussen beide semesters, een voorbeeld hiervan is de Masterproef.

Studiepunten Ieder vak bevat studiepunten, deze beschrijven de geschatte werklast voor dit vak. Vakgroepen vereisen vaak dat je een bepaalde hoeveelheid een studiepunten opneemt.

Predicaten

De predicaten zijn opgedeeld in twee verzamelingen, de eerste omvat alle predicaten die op voorhand gegeven zijn en waarvan de waarheidswaarde al bekend is. In feite zijn dit de predicaten die de eigenlijke samenstelling van een opleiding uitdrukken. Dit is een lijst van de predicaten die tot deze groep behoren:

IsType(VakGroep, VakGroepType) Verzamelingen van vakken (of vakgroepen) behoren altijd tot één bepaald type. Voor elk type van vakgroep geldig bepaalde regels bovenop de regels de mogelijk van toepassing zijn op de vakgroep zelf.

InVakGroep(Vak, VakGroep) Vakken maken altijd deel uit van een bepaalde vakgroep, $\text{InVakGroep}/2$ drukt deze relatie uit.

InFase(Vak, Fase) Dit predicaat drukt uit in welke fases van de opleiding een vak gevolgd kan worden, voor elk vak bestaat er minstens één fase waarin het gegeven wordt. Als de student en vak wil volgen heeft hij/zij de keuze uit één van deze fases.

Verplicht(Vak, Fase) Vakken die behoren tot een vakgroep kunnen mogelijk verplicht, m.a.w. de student zal ze moeten opnemen.

De tweede verzameling zijn de predicaten waarvan de waarheidswaarde op voorhand niet gekend is, en waarvan verwacht wordt dat de gebruiker deze stap per stap invult. Hiertoe behoren slechts twee predicaten namelijk $\text{Geselecteerd}(\text{Vak}, \text{Fase})$ en $\text{GeenInteresse}(\text{Vak})$. $\text{Geselecteerd}/2$ geeft aan welke vakken de student wenst op te nemen tijdens de opleiding. $\text{GeenInteresse}/1$ daarentegen is eigenlijk een predicaat waarmee aangegeven kan worden dat een student helemaal geen interesse heeft om een vak te volgen. Dit is vooral handig als de student een geldige selectie wil laten genereren door het systeem waarbij hij/zij op voorhand kan aangeven dat deze selectie zeker niet de vakken kiest waarin de student geen interesse heeft. Het is niet verplicht dat de gebruiker invulling geeft aan dit predicaat. Door aan te geven dat de student geen interesse heeft om een bepaald vak te volgen kan het systeem afleiden dat er geen fases zijn waarin het vak geselecteerd kan worden.

Functies

De functies koppelen een bepaald resultaat aan een combinatie van waarden uit het domein. Zo geven de functies $\text{MinAantalStudiepunten}(\text{VakGroep}): \text{Studiepunten}$ en

`MaxAantalStudiepunten(VakGroep):Studiepunten` respectievelijk het minimum en het maximum aantal studiepunten weer dat de student verplicht is te selecteren. Deze relaties zijn gekend op voorhand door het systeem en de student moet uiteindelijk trachten zich hieraan te houden. `GeselecteerdAantalStudiepuntenPerVakGroep(VakGroep):Studiepunten` is de relatie tussen het de som van de studiepunten van de vakken die reeds geselecteerd zijn die behoren tot de betreffende vakgroep. Naargelang de selectie kunnen deze waarden verschillen aangezien ze afhankelijk zijn van de keuze van de gebruiker. In het vocabularium zijn meerdere functies terug te vinden die gelijkaardige relaties uitdrukken zoals hierboven beschreven.

2.2 Front-End

Herinner dat het samenstellen van het ISP een interactief configuratieprobleem is, en om aan de eigenschappen van zo'n IC probleem te kunnen voldoen moet de reactietijd van de inferentiemethoden snel genoeg zijn. Met snel genoeg bedoelen we niet meer dan enkele seconden en in sommige gevallen zelfs nog minder. Afhankelijk van hoe frequent sommige methodes gebruikt worden kunnen de eisen omtrent reactietijd verschillen. Om dit te testen heb ik een front-end applicatie ontwikkeld die met behulp van een grafische interface een student toelaat een ISP samen te stellen. Deze input wordt vertaald naar `FO(·)` en doorgespeeld naar IDP. Vervolgens voert IDP inferentie uit en de resultaten hiervan worden terug doorgespeeld naar de front-end, die het resultaat zal weergeven d.m.v. de GUI. Zo kunnen de prestaties van IDP in een gebruiksvriendelijke omgeving getest worden.

2.2.1 Grafische Gebruikersinterface

De interface van de applicatie ziet er gelijkaardig uit aan die van de bestaande web applicatie. Hierdoor kunnen potentiële gebruikers die bekend zijn met de web applicatie snel vertrouwd geraken met het nieuwe systeem.

*/*BESCHRIJVING?*/*

2.2.2 Communicatie tussen Front- en Back-end

Niet alleen is de syntax tussen IDP en Python compleet verschillend, het zijn ook twee verschillende klassen van programmeertalen respectievelijk declaratief en objectgeoriënteerd. Dus communicatie tussen de twee talen is niet vanzelfsprekend. Een Python API die toelaat IDP te gebruiken is reeds ontwikkeld ([Vennekens, 2015](#)). Het doel van de API is om de kloof tussen de declaratieve omgeving van IDP en Python kleiner te maken. Door een populaire programmeertaal te kiezen hoopt de auteur dat informatici met een achtergrond in declaratieve talen en logica sneller van IDP gebruik zullen maken zelfs als ze de syntax van IDP niet meester zijn. Persoonlijk heb ik wel al ervaring met IDP en heb dus geopteerd om geen gebruik te maken van de API.

De input van IDP is compleet tekstueel, de front-end moet een volledig tekstbestand genereren in een formaat dat IDP kan begrijpen. De output van IDP is

opnieuw een tekst-bestand dat de front-end zal moeten parsen om het resultaat te kunnen verwerken. Zo'n tekst-bestand heeft altijd dezelfde opmaak, het bevat een theorie, vocabulary, structuur, procedure en eventueel termen in geval van minimalisatie. De front-end bevat een parser object dat voor elke oproep zo'n tekst bestand zal opbouwen. De theorie, vocabulary en de procedures zijn altijd dezelfde en hoeven slecht éénmalig ingelezen te worden. Enkel de structuur zal bij elke nieuwe actie gedeeltelijk dynamisch gegenereerd moeten worden. De predicaten in het vocabulary zijn opgedeeld in twee verzamelingen. De eerste verzameling Γ bevat de predicaten die vooraf gegeven zijn en gekend zijn door het systeem. Anders gezegd bestaat er een interpretatie G voor Γ . De andere verzameling Ω , zijn de predicaten waar nog geen interpretatie W voor gegeven is en die door de gebruiker (in dit geval de student) ingevuld zullen moeten worden. Tot deze laatste groep behoren volgende predicaten: Geselecteerd(Vak,Fase) en NietGeïnteresseerd(Vak). De bedoeling is om een invulling te zoeken zodat $W \cup G$ consistent is met alle regels van de theorie T .

2.3 Features

2.3.1 Selectieproces

Elke keuze die de gebruiker maakt moet zorgvuldig afgehandeld worden om te kunnen geranderen dat de selectie altijd satisfieerbaar is. De voorwaarde is dat als de gebruiker een keuze maakt die ervoor zorgt dat de theorie niet langer satisfieerbaar is, hij hier meteen op de hoogte van wordt gebracht zodanig dat hij dit kan oplossen alvorens verder te gaan. Er wordt gestart met de minst precieze interpretatie W_0 om uiteindelijk stapsgewijs tot een exacte interpretatie W_n met maximale precisie te komen waarvoor geldt dat $W_n \cup G \models T$. Hoe deze stappen verlopen staat beschreven in onderstaand algoritme. Hier wordt W opnieuw opgedeeld in drie verzamelingen U , P en O . U is de verzameling van predicaten met een waarheidswaarde $\geq P$ u , waarvoor geldt dat deze toegekend zijn door de gebruiker. De tweede verzameling P bevat eveneens predicaten met een waarheidswaarde $\geq_P u$, maar deze zet zijn de propagaties die volgen uit U . O tenslotte is de set van predicaten waarvoor de waarheidswaarde nog onbekend is. Voor W_0 geldt: $W_0 = O_0$ en $U_0 = P_0 = \emptyset$. In het verder verloop van de tekst zal deze beschrijven blijven gelden en terugkomen.

2.3.2 Propagatie

Bij het samenstellen van een ISP kan het systeem de informatie die de gebruiker reeds verder gaan propageren. Hierbij baseert het systeem zich op de regels van de theorie. Neem het volgende voorbeeld:

$$\text{Geselecteerd}(A) \Rightarrow \text{Geselecteerd}(B)$$

Deze regel zegt dat als A geselecteerd is, B ook geselecteerd moet zijn. Als de gebruiker A selecteerd, zal het systeem hieruit afleiden dat B ook geselecteerd moet doen. In de GUI zal te zien zijn dat B ook geselecteerd is, hoewel de gebruiker dit niet expliciet heeft gekozen.

Algorithm 1: Selectieproces

```

1 function Selectieproces ( $U_i$ );
   Input : De nieuwe keuze van de gebruiker  $P_i$ 
   Output :
2 if  $\text{Sat}(U_i)$  then
3    $V \leftarrow P_{i-1} \cap O_i$ ;
4   if  $V \neq \emptyset$  then
5      $V' \leftarrow \text{KeuzeGebruiker}(V)$ ;
6     Selectieproces( $U_i \cup V'$ );
7   else
8      $P_i \leftarrow \text{Propagate}(U_i)$ ;
9     NieuweGebruikerActie( $U_{i-1}, U_i, P_{i-1}, P_i, O_{i-1}, O_i$ );
10  end
11 else
12   $N \leftarrow \text{Unsat}(U_i)$ ;
13   $K, L \leftarrow \text{ManueleResolutie}(N)$ ;
14   $U'_i \leftarrow U_i [K] - L$ ;
15  Selectieproces( $U'_i$ );
16 end

```

2.3.3 Model Expansie

Als de gebruiker al zijn/haar voorkeuren heeft ingevuld in de partiële interpretatie W_i met $U_i \cup P_i \cup O_i = W_i$ en $O_i \neq \emptyset$. Dan kan het systeem de verzameling van predicaten O_i waarvan de waarheidswaarde nog onbekend is, verder oplossen. Om zo tot een interpretatie W_f te komen met $U_i \cup P_i \subseteq W_f$. W_f is tevens (precisie) maximaal en $W_f \cup G \models T$. Anders gezegd zal het systeem de partiële selectie verder vervolledigen tot een geldig ISP.

2.3.4 Minimизatie

Het is niet alleen mogelijk om een partiële selectie verder te laten vervolledigen, maar om hierbij ook een parameter in acht te nemen en een interpretatie W_f te bekomen waarbij deze parameter zo klein mogelijk is. Het systeem kan een optimalisatie van volgende parameters zoeken:

Werklast Met werklast wordt bedoeld het aantal studiepunten dat opgenomen wordt. Een student is verplicht vakken te selecteren en gekoppeld aan die vakken zijn studiepunten. De student kan vragen aan het systeem om een ISP samen te stellen waarbij de werklast (de som van de studiepunten van de geselecteerde vakken) zo laag mogelijk is.

Werklast per semester Vakken die geen jaarvakken zijn, vallen ofwel in het eerste semester of het tweede. Het systeem kan een ISP samenstellen waarbij het verschil in studiepunten

tussen geselecteerde vakken van eerste en tweede semester zo klein mogelijk is. Zodat de werklast zo goed mogelijk verdeeld is tussen beide semesters.

Overlap Vakken hebben uiteraard lesmomenten en het kan al eens gebeuren dat deze lesmomenten voor verschillende vakken samenvallen. Dit is uiteraard niet ideaal voor de student om die niet op twee plaatsen tegelijk kan zijn. Stel dat er het lesmoment van vak A gepland is van 16u tot 18u en het lesmoment voor vak B vindt plaats op dezelfde dag van 15u tot 17u. De overlap bedraagt dan 60 minuten. De student kan een ISP laten genereren waarbij de totale som van alle overlap zo minimaal mogelijk is.

2.3.5 Ongedaan Maken

Het ongedaan kunnen maken van selecties is een van de aspecten waar meerdere strategieën mogelijk zijn. Elke stap in het selectieproces (nieuwe selectie door de gebruiker) wordt bijgehouden in een zogenaamde actie. Deze actie bevat niet alleen die nieuwe interpretatie W_i maar ook de voorgaande W_{i-1} zodanig dat als een actie ongedaan gemaakt wordt het systeem weet wat de toestand voorheen was en hier naar kan terugkeren. Het is niet noodzakelijk zo dat in een actie A_i met interpretaties W_{i-1} en W_i dat $W_{i-1} \leq_p W_i$. Acties beschrijven elke handeling in de tijd gemaakt door de gebruiker, dus ook de waarheidswaarde van een predicaat minder specifiek maken. Ongedaan maken in deze context betekent dus niet automatisch de huidige selectie minder specifiek maken, maar eerder terugkeren in de tijd.

Strategie 1a De eerste strategie houdt in dat als een actie ongedaan gemaakt wordt de interpretatie W_i simpelweg vervangen wordt door de voorgaande W_{i-1} . Dit is wellicht de meest simpele strategie aangezien er geen extra bewerkingen uitgevoerd moeten worden buiten het terugkeren naar de voorgaande interpretatie.

Algorithm 2: MaakActieOngedaan

```

1 function MaakActieOngedaan (Ac);
   Input : De actie die ongedaan gemaakt moet worden  $A_i$ 
   Output :
2 verwijder( $A_i$ );
3  $W_i \leftarrow W_{i-1}$ ;
```

Strategie 1b Bij de voorgaande strategie staat een actie ongedaan maken gelijk aan simpelweg de klok terugdraaien en W_i vervangen door W_{i-1} . Dus eventuele propagaties die volgden uit de keuze van de gebruiker in de actie worden ook ongedaan gemaakt. En in plaats van alles simpelweg ongedaan te maken zou het beter zijn als de gebruiker de keuze krijgt om eventuele propagaties te behouden. Dit zorgt echter voor het volgende probleem. De assumptie is dat acties handelingen weergeven in de tijd gemaakt door de gebruiker, het ongedaan maken van handelingen wordt dus

gezien als het terugdraaien van de tijd. Maar als de gebruiker kiest om een actie ongedaan te maken en tegelijk eventuele propagaties toch wil behouden dan geldt deze assumptie niet meer. In ?? staat beschreven hoe dit probleem opgelost kan worden. Het is simpelweg een uitbreiding op de eerste strategie, een actie wordt nog altijd verwijderd en de laatste interpretatie W_i wordt vervangen door de voorgaande W_{i-1} . Maar waar het hier voorheen stopte gebeurt er nu het volgende. Er wordt gecontroleerd of er een set van predicaten V bestaat die zowel toebehoren aan O_{i-1} en P_i . Ofwel predicaten waarvoor de waarheidswaarde eerst onbekend was en daarna een meer specifieke waarheidswaarde hebben gekregen d.m.v. propagatie als gevolg van de keuze van de gebruiker. Is deze set niet leeg dan zal de gebruiker moeten kiezen in KeuzeGebruiker welke van de propagaties te houden. In het geval dat de gebruiker kiest om propagaties te behouden, worden deze niet langer gezien als propagaties maar eerder als keuzes gemaakt door de gebruiker. Deze worden toegevoegd aan de verzameling U en dit wordt gezien als een nieuwe handeling in het selectieproces waarvoor een nieuwe actie gecreëerd zal worden.

Algorithm 3: MaakActieOngedaan

```

1 function MaakActieOngedaan (Ac);
   Input : De actie die ongedaan gemaakt moet worden  $A_i$ 
   Output :
2  $V \leftarrow O_{i-1} \cap P_i$ ;
3 verwijder( $A_i$ );
4  $W_i \leftarrow W_{i-1}$ ;
5 if  $V \neq \emptyset$  then
6    $V' \leftarrow \text{KeuzeGebruiker}(V)$ ;
7   if  $V' \neq \emptyset$  then
8     Selectieproces( $U_{i-1} \cup V'$ );
9   end
10 end

```

Strategie 2 Voorheen was de assumptie dat een actie ongedaan maken gelijk stond aan de tijd als het ware terugdraaien en de interpretatie vervangen door de voorgaande. En dat dit niets te maken had met de precisie ordening van de interpretaties. Voor deze strategie wordt er niet meer uitgegaan van deze assumptie, maar in plaats daarvan staat ongedaan maken gelijk aan eender welke keuze van de gebruiker die ervoor zorgt dat de interpretatie minder precies wordt. Dit kan zijn de waarheidswaarde van een predicaat ongedaan (minder precies) maken of een waarheidswaarde van een predicaat veranderen van bv. waar naar onwaar. Met als gevolg dat de interpretatie minder precies is omdat propagaties niet meer gelden. Wat verandert is dat in deze situatie geen expliciete acties meer worden bijgehouden zoals voorheen, en dat ongedaan maken nu een onderdeel is van het selectieproces zelf. Kort gezegd verandert enkel de assumptie over ongedaan maken. De rest van

de werking blijft dezelfde, behalve dat zoals voorheen niet meer gebruikt worden. De werking staat beschreven in 4.

Algorithm 4: Selectieproces

```

1 function Selectieproces ( $U_i$ );
   Input : De nieuwe keuze van de gebruiker  $P_i$ 
   Output :
2 if  $\text{Sat}(U_i)$  then
3    $V \leftarrow P_{i-1} \cap O_i$ ;
4   if  $V \neq \emptyset$  then
5      $V' \leftarrow \text{KeuzeGebruiker}(V)$ ;
6     Selectieproces( $U_i \cup V'$ );
7   else
8      $P_i \leftarrow \text{Propagate}(U_i)$ ;
9     NieuweGebruikerActie( $U_{i-1}, U_i, P_{i-1}, P_i, O_{i-1}, O_i$ );
10  end
11 else
12    $N \leftarrow \text{Unsat}(U_i)$ ;
13    $K, L \leftarrow \text{ManueleResolutie}(N)$ ;
14    $U'_i \leftarrow U_i [K] - L$ ;
15   Selectieproces( $U'_i$ );
16 end

```

2.4 Conflict Explanation

Het is mogelijk dat de gebruiker een verkeerde keuze maakt waardoor de theorie niet meer satisfieerbaar is. Simpel gezegd heeft de gebruiker een waarde gekozen voor een bepaalde variabele, zodanig dat dit ervoor zorgt dat één of meerdere regels uit de theorie samen niet meer waar kunnen worden gemaakt.

$$\text{Geselecteerd}(A) \wedge \text{Geselecteerd}(B)$$

Het bovenstaande voorbeeld van een theorie bevat één regel die zegt dat A en B beide geselecteerd moeten zijn. Stel dat de gebruiker nu kiest $\neg \text{Geselecteerd}(A)$. Hieruit volgt dat de regel uit de theorie nooit waar kan zijn, of niet satisfieerbaar wordt. Dit is een conflict en het is aan het systeem om een de gebruiker duidelijk te maken dat er een probleem is, wat het probleem inhoudt en hoe de gebruiker dit kan oplossen. Dit laatste behoort tot het domein van conflict explanation. IDP gebruikt momenteel twee technieken om oorzaken van niet satisfieerbaarheid op te sporen. De unsatstructure spoort een set van variabelen op die het probleem veroorzaken. Vervolgens is er de unsattheory, die zoekt naar een minimale set van regels uit de theorie die niet waar gemaakt kunnen worden gegeven de huidige selectie.

Unsatstructure

De unsatstructure is een efficiënte tool om aan de gebruiker te kunnen meedelen, welke van de voormalige selecties problemen veroorzaken. In het voorbeeld van het ISP werd er een opsplitsing gemaakt tussen twee sets van predicaten, Γ waarvoor de invulling G vooraf bekend is en Ω waarvoor een interpretatie W samengesteld dient te worden door de student. W bestond zelf opnieuw uit drie verzamelingen U , P en O . Als een interpretatie ervoor zorgt dat de theorie niet meer satisfieerbaar wordt, dan is dit te wijten aan de keuze van de gebruiker U . En het is van deze verzameling dat de unsatstructure een precisie minimale deelverzameling V zal zoeken als oorzaak van het probleem. De student krijgt een oplistijng te zien van V , en zal de mogelijkheid krijgen om hier veranderingen in aan te brengen zodoende het probleem op te lossen.

Reified Constraints

Wat opvalt is dat de voorgaande techniek enkel een verzameling selecties teruggeeft die bijdragen tot het probleem. Maar verdere uitleg over welke regel(s) uit de theorie niet meer waar gemaakt kunnen worden en waarom wordt niet gegeven. Dus de gebruiker krijgt enkel de foute selectie te zien, zonder erbij te zeggen wat er juist mis mee is. Herinner de unsattheory zoekt achter een minimale set van regels uit de theorie die niet meer waar gemaakt kunnen worden. Dit is het exact hetgene wat de gebruiker hoort te weten om het probleem op te lossen, maar het probleem zit het hem in de formulering. De unsattheory formuleert de fouten in de IDP syntax $FO(\cdot)$. Deze syntax is voor de doorsnee informaticus met een achtergrond in eerste orde logica of IDP goed te lezen, maar voor een student die geen ervaring heeft met deze domeinen zal ongetwijfeld niet in staat zijn deze regels te ontcijferen. Neem als voorbeeld onderstaande regel, elke student computerwetenschappen zal vrij snel kunnen achterhalen wat de regel inhoudt. Maar personen zonder deze achtergrond zullen dit niet kunnen verstaan.

LISTING 2.1: IDP Rule Example

$$\begin{aligned} \forall vg [VakGroep] : & \text{IsType}(vg, AVO) \Rightarrow \text{GesAantalStupunVakGr}(vg) \\ = & \text{sum}\{v[Vak], \text{sp}[Studiepunten], f[Fase] : \text{InVakGroep}(v, vg) \\ \wedge & \text{Geselecteerd}(v, f) \wedge \text{AantalStudiepunten}(v) = \text{sp} : \text{sp}\}. \end{aligned}$$

Een mogelijke oplossing voor dit probleem komt in de vorm van reified constraints. De theorie T bestaat uit regels (constraints) en hierin wordt onderscheid gemaakt tussen 2 sets van regels. De eerste is de set van regels B die altijd waar zullen zijn, deze noemen we achtergrond constraints. De andere set is die van regels F genaamd voorgrond constraints die de gebruiker moet zien waar te maken d.m.v. een correcte interpretatie samen te stellen. Het is deze set van regels F , waarvoor reified constraints gebruikt zullen worden. Voor zij die niet bekend zijn met het concept van reified constraints, het is een zeer simpele techniek. Een regel C (constraint) kan heel simpel omgevormd worden tot een reified constraint door C equivalent te maken aan een booleaanse waarde B . Dus een regel C ziet er als dan als volgt uit, $B \leftrightarrow C$. De waarheidswaarde van C is dan equivalent aan die van B , dus als de regel

niet consistent of 'false' is, zal B dus ook 'false' zijn, hetzelfde geldt ook voor 'true' uiteraard. De voorgaande regel ziet er dan als volgt uit.

LISTING 2.2: IDP Reified Constraint Example

$\begin{aligned} \text{Regel} \leftrightarrow \forall \text{vg} [\text{VakGroep}] : & \text{IsType}(\text{vg}, \text{AVO}) \Rightarrow \text{GesAantalStupunVakGr}(\text{vg}) \\ = \text{sum}\{v[\text{Vak}], \text{sp}[\text{Studiepunten}], f[\text{Fase}] : & \text{InVakGroep}(v, \text{vg}) \\ \wedge \text{Geselecteerd}(v, f) \wedge \text{AantalStudiepunten}(v) = \text{sp} : & \text{sp} \}. \end{aligned}$

Rule is de booleaanse variabele wiens waarheidswaarde die van de regel reflecteert. Op deze manier hoeven we enkel na te gaan of Rule true of false is om te weten te komen of de regel (in)consistent is gegeven de selectie van de gebruiker. Dit alleen biedt natuurlijk niet veel extra informatie, maar het nut van de booleaanse variabele is wel degelijk belangrijk. Voor elke regel uit F is er op voorhand een beschrijving opgesteld in natuurlijke taal met de redenering achter het eventuele falen van die regel. Aan de hand van de reified constraints kan er heel gemakkelijk achterhaald worden welke regels er wel of niet consistent zijn door simpelweg te kijken naar de booleaanse waarde in de regels. Is de booleaanse variabele false, dan moet de gebruiker de bijhorende beschrijving te zien krijgen voor deze regel. Hoewel dit een zeer gemakkelijke en generische manier is om aan conflict explanation te doen, zijn de mogelijkheden van deze techniek toch gelimiteerd en niet altijd voldoende toereikend. Ten eerste is de uitleg vaak oppervlakkig en niet specifiek genoeg om duidelijk te kunnen verklaren waarom een regel inconsistent is.

LISTING 2.3: Reified constraint Shortcomings

$\begin{aligned} \forall \text{vg} [\text{VakGroep}] : & \text{MaxAantalStudiepunten}(\text{vg}) > 0 \Rightarrow \\ \text{MinAantalStudiepunten}(\text{vg}) \leq & \text{GeselecteerdAantalStudiepuntenPerVakGroep}(\text{vg}) \leq \text{Ma} \end{aligned}$
--

Bovenstaande regel stelt dat voor elke vakgroep, het totaal aantal geselecteerde studiepunten van deze vakgroep tussen diens minimum en maximum moet liggen, als het maximum aantal studiepunten ervan groter is dan nul. In het geval dat deze regel inconsistent is krijgt de student de volgende uitleg te zien. Je hebt niet voor alle vakgroepen tussen het minimum en maximum aantal studiepunten geselecteerd. De verklaring is algemeen en geeft geen specifieke details. Zo zegt het niet voor welke vakgroep het geselecteerd aantal studiepunten niet klopt, en of het er nu teveel of te weinig zijn. Dus de verklaringen zijn redelijk beperkt in hun mogelijkheden. En ten tweede laat deze methode enkel toe om inconsistente regels op te sporen. Een Interactief Configuratieprobleem zoals het ISP bevat een verzameling predicaten Ω waarvoor de gebruiker (in dit geval een student) stapsgewijs een interpretatie W samenstelt. Deze interpretatie moet uiteindelijk als ze compleet is, een geldig model zijn volgens de regels van de theorie. Een regel is inconsistent als ze niet voldaan is gegeven de huidige partiële selectie. Dit wil niet zeggen dat de regel niet meer consistent kan worden naarmate de interpretatie verder ingevuld wordt door de student. Terwijl een set van regels die nooit samen waar gemaakt kunnen worden gegeven de huidige partiële selectie niet-satisfieerbaar zijn. Het is deze laatste set van regels die dient gevonden te worden. De techniek van reified constraints op zich kan dit niet verwezenlijken. Maar door deze techniek te combineren met de

unsatstructure wordt dit wel mogelijk. Hiervoor moet er een nieuwe theorie T' opgesteld worden. Herinner dat de regels uit T opgedeeld zijn in twee sets, B de achtergrond constraints en F de voorgrond constraints. T' bevat ook beide sets maar de regels uit F zijn omgevormd tot reified constraints. /*INSERT EXAMPLE CODE*/ Het vocabularium voor T' is dan ook hetzelfde als dat van T met de toevoeging van het predicaat $\text{Regel}(\text{Nummer})$. Dit stelt de waarheidswaarde van een reified constraint voor, waarbij elke regel een uniek nummer krijgt. Nemen we terug de regel uit $??$, dan ziet deze er als volgt uit:

LISTING 2.4: IDP Reified Constraint Example 2

$$\begin{aligned} \text{Regel}(1) \leftrightarrow \forall vg[VakGroep] : \text{IsType}(vg, AVO) \Rightarrow \text{GesAantalStupun} & \text{VakGr}(vg) \\ = \text{sum}\{v[Vak], \text{sp}[Studiepunten], f[Fase] : \text{InVakGroep}(v, vg) & \\ \wedge \text{Geselecteerd}(v, f) \wedge \text{AantalStudiepunten}(v) = \text{sp} : \text{sp} \} & . \end{aligned}$$

Belangrijk is dat de waarheidswaarde van dit predicaat voor elke regel op 'true' staat, dit wil zeggen dat ervan uit wordt gegaan dat aan elke regel voldaan moet worden. Wanneer blijkt dat een selectie U ervoor zorgt dat de theorie T niet meer satisfieerbaar is, wordt deze gebruikt samen met T' . Sinds ervan uit wordt gegaan dat elke regel waar moet zijn ($\text{Regel}(\text{Nummer})$ is true voor elke regel) zal er dus nooit een invulling bestaan die U bevat die dat kan waarmaken. Maar i.p.v. de unsatstructure te laten zoeken naar een precisie minimale subset van U , gaat deze opzoek naar een precisie minimale subset van de verzameling predicaten $\text{Regel}(\text{Nummer})$. Deze set is een verzameling van regels die niet-satisfieerbaar zijn volgens U .

Amilhastre

/*TITEL?*/ De unsatstructure is zeker een belangrijk onderdeel van het IDP systeem. En hoewel de techniek zeker nuttig is, schiet ze toch wat tekort. In geval van een niet-satisfieerbare selectie U , zal de methode op zoek gaan naar één precisie minimale subset $V \subseteq U$, die niet-satisfieerbaar is. Met andere woorden wijst het de oorzaak van het probleem aan. Dit is een stap in de juiste richting, maar nog beter zou zijn als niet enkel de oorzaak wordt aangewezen maar ook alle mogelijk oplossingen voor dit probleem. Met oplossing wordt bedoeld de kleinste verzameling van predicaten die ongedaan gemaakt dienen te worden om het probleem terug satisfieerbaar te maken.

In (?) wordt getoond hoe de oplossingsverzameling van een CSP voorgesteld kan worden door een eindige toestandsautomaat. Deze automaat, die vooraf berekend wordt kan vervolgens gebruikt worden om tijdens het selectieproces de complexiteit van de bewerkingen van verscheidene inferentie taken te reduceren en zo de reactietijd aanzienlijk te verbeteren. Eén van deze taken is het voorzien van oplossingen in geval van een niet-satisfieerbare selectie, en dit is exact hetgeen waar naar we op zoek zijn. De focus van de paper ligt op het reduceren van de complexiteit van de bewerkingen gedurende het selectieproces met behulp van de vooraf berekende automaat. En hoewel dit niet de focus is van dit onderzoek, biedt de techniek wel de inferentie taak aan die we zoeken. En met de garantie, zo stelt de paper dat de complexiteit van de bewerkingen aanzienlijk lager ligt dan normaal.

De automaat is een compacte voorstelling van de oplossingsverzameling van een configuratieprobleem. Elk pad is een combinatie van waarden uit het domein van de variabelen die samen een correcte interpretatie vormen volgens de theorie. De voorstelling van IDP is echter anders, hierbij geeft de gebruiker invulling d.m.v. toekenning van waarheidswaarden aan de verzameling van predicaten Geselecteerd(Vak,Fase). Als de student dus waarheidswaarde true toekent aan het predicaat Geselecteerd(G0Q66C,2), betekent dit dat de student het vak 'Fundamentele van Computergrafiek' wil volgen in de 2de fase van de opleiding. Door elk uniek vak voor te stellen als variabele met als domein de mogelijke fases waarin ze gegeven worden krijgen we een formaat dat nodig is voor de automaat. Voor elk vak dient 0 toegevoegd te worden aan het domein, dit voor wanneer een vak niet geselecteerd is door de gebruiker.

LISTING 2.5: IDP Syntax conversion

$$\begin{aligned} \text{Vak} &= \{G0Q66C; \dots\} \rightarrow V_{G0Q66C} \\ \text{InFase} &= \{G0Q66C, 1; G0Q66C, 2; \dots\} \rightarrow D_{V_{G0Q66C}} = \{0, 1, 2\} \end{aligned}$$

Via dit formaat van conversie is het dus mogelijk om de automaat te construeren en er mee te communiceren. In geval van een niet-satisfieerbare selectie zal de techniek volgens Amilhastre alle minimale oplossingen zoeken voor het probleem. Nog een voordeel van de automaat is dat ze werkt op basis van gewichten. Standaard heeft elk vak dezelfde voorkeur (gewicht) van 1, hierdoor is een minimale oplossing een oplossing waarbij zo weinig mogelijk keuzes ongedaan gemaakt worden. Maar door de gewichten in de automaat aan te passen kunnen voorkeuren mee in rekening worden gebracht, dit zal een effect hebben op de minimale oplossingen. Het aanpassen van deze gewichten vereist geen herberekening van de automaat en is ook geen kostelijke operatie. /*INSERT SMALL EXAMPLE*/

Chapter 3

Evaluation

De bedoeling van dit onderzoek is om na te gaan of de regels van het ISP in IDP beschreven kunnen worden. En hoe algemeen deze regels gemaakt kunnen worden zodanig dat dezelfde theorie meerdere opleidingen kan beschrijven. Daarnaast is het de bedoeling om het lessenrooster mee in rekening te brengen. Aangezien het mogelijk is dat vakken overlappen, is het toch belangrijk dat de student op de hoogte is van deze informatie. Met deze regels willen we natuurlijk inferentie kunnen doen, en een belangrijke vereiste van een Interactief configuratieprobleem zoals een ISP samenstellen is een snelle respons. De inferentietaken moeten dusdanig snel afgehandeld kunnen worden dat de reactietijd hooguit enkel seconden bedraagt. En tenslotte is er nog conflict explanation, waarbij ik wil nagaan wat de mogelijkheden zijn binnen ISP en of andere externe technieken ? in samenwerking met IDP extra mogelijkheden kunnen bieden.

3.0.1 Theorie van het ISP

Geen twee opleidingen zijn dezelfde, en toch zou het gemakkelijk zijn als ongeacht deze verschillen we telkens dezelfde set van regels kunnen gebruiken zonder daarin te moeten gaan aanpassen. Om de omvang van het project haalbaar te houden voor één persoon, is het domein van opleidingen beperkt gebleven tot die van de computerwetenschappen en informatica. Hiervoor ben ik erin geslaagd en theorie op te stellen die zonder probleem eender welk van deze opleidingen kan beschrijven. Hoewel de naamgeving vaak verschilt per opleiding, komen toch vaak structuren voor die dezelfde eigenschappen vertonen. Dus de echte uitdaging is deze structuren vinden, niet de regels ervoor schrijven. Voor een klein sub-domein binnen de opleidingen van de KU Leuven is het dus mogelijk een theorie te vinden.

3.0.2 Front-End Applicatie

Het hoofddoel van de Front-End (incl. Grafische User-Interface) is het simuleren van een mogelijke toekomstige applicatie die effectief gebruikt kan worden door studenten en om gemakkelijk de inferentie technieken van het IDP systeem te kunnen testen in een real-time setting. Momenteel bestaat er al een web applicatie waarmee een

TABLE 3.1: Functionaliteiten

	Web applicatie	Nieuwe applicatie
Handmatige selectie	✓	✓
Controle correctheid	✓	✓
Verklaring v. incorrectheid	✓	✓
ISP genereren		✓
Optimaal ISP genereren		✓
Propagatie		✓
Ongedaan maken	✓	✓
Geschiedenis v. Acties		✓
Weergave lessenrooster		✓

student een ISP kan samenstellen. Deze kan als maatstaf dienen in de beoordeling van het nieuwe systeem. Er kan namelijk pas gesproken worden van vooruitgang als de nieuwe toepassing betere prestaties levert en uitgebreidere functionaliteit biedt ten opzichte van de reeds bestaande standaard. Hierbij ligt de focus op (i) het verschil in functionaliteiten (welke functionaliteit is aanwezig in welke applicatie?) en (ii) het niveau van ondersteuning dat ze aanbieden. Criteria zoals design en platform zullen hier niet aan bod komen aangezien deze niet tot de essentie van dit onderzoek behoren. Prestaties van de verschillende inferentie technieken staan besproken in een verder hoofdstuk.

De huidige web applicatie van de K.U. Leuven biedt een basispakket van ondersteuning aan, essentieel voor de student om zijn/haar ISP te kunnen vervolledigen. Het laat de student toe om te kiezen welke opleidingsonderdelen van de gekozen opleiding hij/zij van plan is te volgen. De selectie kan vervolgens gecontroleerd worden op correctie, m.a.w. het systeem zal nagaan of ze voldoet aan alle regels van het ISP. En als de selectie niet voldoet aan deze regels zal het systeem dit melden en tevens ook uitleg voorzien waarom de selectie niet voldoet aan de regels. Deze functionaliteiten zitten ook allemaal inbegrepen in de nieuwe applicatie, samen met nog een aantal nieuwe functies. Hiermee kan ik veilig stellen dat het aanbod van functionaliteit in mijn nieuwe applicatie zeker uitgebreider is dan dat van het huidige systeem. Naast het basispakket is het ook mogelijk voor de gebruiker om een partiële selectie verder te laten invullen door het systeem tot het een volledig geldige selectie is volgens de regels van het ISP, het is zelfs mogelijk om een optimaal ISP te laten opstellen volgens een bepaald criterium. Gedurende het selectieproces is het mogelijk dat de keuzes van gebruiker ervoor zorgen dat bepaalde opleidingsonderdelen geselecteerd moeten worden als een gevolg hiervan, het systeem zal deze keuzes automatisch invullen (propageren) zodat de gebruiker zich hier geen zorgen over moet maken. Elke nieuwe stap in het selectieproces wordt bijgehouden in een geschiedenis van acties, als de gebruiker niet tevreden is met de selectie en deze wil wissen dan kan dit door de acties ongedaan te maken. Wanneer de student een vak selecteert dan zal het systeem een partieel lessenrooster genereren, dit is handig om te controleren of mogelijk geen lesmomenten overlappen met elkaar.

Er is dus een duidelijk verschil in het aanbod van functionaliteiten, maar bieden ze ook allemaal hetzelfde niveau van ondersteuning? Om te beginnen zullen we de gemeenschappelijke onderdelen onder de loep nemen. Als eerste is er de handmatige selectie, wellicht de meest essentiële van alle functionaliteiten. In beide applicaties geeft het de student de mogelijkheid om aan te vinken welke van de vakken hij/zij wenst te volgen, beide applicaties werken op dezelfde manier met uitzondering dat het in de web applicatie niet mogelijk is om aan te geven dat je niet geïnteresseerd bent in een vak.

Zowel de bestaande als de nieuwe app kunnen een selectie controleren op geldigheid, maar de nieuwe applicatie doet dit bij elke nieuwe selectie van de student zodanig dat fouten zo snel mogelijk gedetecteerd worden. Bij de web applicatie gebeurt dit enkel wanneer de student denkt klaar te zijn en diens keuze bevestigt.

Blijkt dan dat de selectie niet geldig is volgens de regels, dan dient hier een verklaring voor gegeven te worden. Het bestaande systeem biedt duidelijke verklaringen in de zin dat het kan verklaren welke regels er niet kloppen. De duidelijkheid van de verklaringen in het nieuwe systeem is sterk afhankelijk van de IDP theorie die het gebruikt, de opmerking werd al gemaakt dat een algemene theorie niet geschikt is om de gebruiker goed te kunnen informeren i.p.v. een specifieke theorie. Het is dus mogelijk voor de nieuwe applicatie om evenveel duidelijkheid te kunnen scheppen als de web applicatie, maar dit zal sterk afhangen van de gebruikte theorie en dus ten koste zijn van andere factoren. Maar hier houdt het voor de web applicatie op, naast tekstuele verklaringen van de regels die verbroken zijn biedt het systeem geen verdere ondersteuning die de gebruiker kunnen helpen bij het oplossen van een foute selectie. De nieuwe applicatie kan opsporen in welk deel van de selectie de fout zich bevindt, en is zelfs in staat om alle mogelijk (minimale) oplossingen te tonen aan de gebruiker.

Het bijhouden van de acties gemaakt door de gebruiker in een geschiedenis is ook één van de nieuwe features. Voorheen was het al mogelijk om selecties ongedaan te maken door simpelweg het vinkje in de checkbox van een geselecteerd vak uit te vinken. Dit is nog altijd mogelijk maar daarbij is het ook nog eens mogelijk om de acties uit de geschiedenis ongedaan te maken. Propagaties die niet meer gelden als gevolg van acties of selecties ongedaan maken zullen getoond worden aan de student die dan kan kiezen of hij/zij deze nog wil behouden of niet.

En tenslotte kan de student een partiële selectie verder laten vervolledigen door het systeem. Hierbij zal het systeem uiteraard de reeds gemaakte keuzes behouden en ze verder vervolledigen. De student kan aangeven als hij/zij niet geïnteresseerd is in bepaalde vakken en bij het verder aanvullen van de selectie zal de nieuwe applicatie deze vakken niet kiezen. Niet alleen is het mogelijk om een partiële selectie verder aan te vullen, maar de student kan als hij/zij dit wenst een optimaal ISP samen laten stellen volgens een bepaalde parameter (bv. de werklast zo gelijk mogelijk verdelen over beide semesters).

3.0.3 Inferentie

Met behulp van de interface kan de gebruiker een ISP samenstellen. Het is de bedoeling om de gebruiker bij te staan in dit proces, dit door gevolgen van bepaalde keuzes door te voeren, een onvolledige selectie te vervolledigen, een optimaal ISP samen te stellen volgens een bepaald criterium, foute keuzes te detecteren en de gebruiker hiervan op de hoogte te brengen etc. En niet te vergeten, al deze processen moeten in real-time gebeuren en dus zeer snel afgehandeld kunnen worden. Binnen het domein van de opleidingen, ben ik tot de conclusie gekomen dat zowat al deze inferentie taken snel zeer snel en efficiënt uitgevoerd worden. De reactietijd, meegerekend het genereren van de IDP text file en het resultaat terug ontcijferen bedraagt in bijna alle gevallen minder dan 1 seconde. Zelfs de resultaten van minimalisatie opdrachten geven goede resultaten terug met een maximale rekentijd van ongeveer 10 seconden.

3.0.4 Conflict Explanation

Voor IC problemen is het de verantwoordelijkheid van het systeem om de gebruiker zo goed mogelijk bij te staan terwijl deze het probleem probeert op te lossen. Dit geldt zeker voor situaties waarbij de gebruiker een keuze heeft gemaakt die nooit tot een geldige oplossing kan leiden. Ik heb in deze paper drie interessante technieken geïmplementeerd die de gebruiker kunnen helpen om een conflict volgend uit een foute keuze op te lossen.

Unsatstructure

De unsatstructure van IDP gaat op zoek naar een precisie minimale subset $V \subseteq U$ (U is de verzameling van alle keuzes gemaakt door de gebruiker) die nooit deel kan uitmaken van een geldig model. Kortom de gebruiker krijgt een lijst te zien met vakken waarvoor hij/zij een keuze heeft gemaakt die de niet-satisfieerbaarheid veroorzaken. Het is in deze lijst dat de gebruiker keuzes kan veranderen of ongedaan maken om dit probleem op te lossen. De werking is zoals verwacht een de reactietijd ligt binnen de eisen van een IC probleem. Enkel een lijst tonen is natuurlijk niet genoeg, als er geen uitleg gegeven wordt waarom de huidige keuze fout is kan je ook niet weten wat je moet veranderen om het terug op te lossen.

Reified Constraints

De techniek van reified constraints biedt extra informatie en kan dienen als een helpende hand om de gebruiker door het selectieproces te loodsen. Deze simpele generische manier om niet-satisfieerbare regels op te sporen en te verklaren in natuurlijke taal heeft zeker zijn voordelen. De implementatiekost ligt zeer laag, maar de functionaliteit van deze techniek is helaas ook beperkt. Door de techniek te combineren met de unsatstructure is het mogelijk om niet-satisfieerbare regels efficiënt op te sporen. En door aan elke regel een uitleg in natuurlijke taal te koppelen (deze hoeft slechts éénmalig per regel opgesteld te worden) kan het systeem de

gebruiker een beter begrijpbare ondersteuning bieden door precies te zeggen welke regel(s) er verbroken is (zijn) en te specificeren hoe dit zou kunnen gebeurd zijn. Maar deze uitleg is zeer statisch, dus als dezelfde regel door verschillende keuzes verbroken wordt zal de uitleg in beide gevallen dezelfde zijn. In vele gevallen is deze informatie duidelijk en specifiek genoeg, maar in de implementatie staat ook een voorbeeld vermeld waarbij de uitleg niet veel extra hulp biedt omdat de regel die ze beschrijft te algemeen is. Sinds het de bedoeling is van dit onderzoek om een zo algemeen mogelijke theorie te ontwikkelen kan de vraag gesteld worden hoe nuttig het is om deze techniek te gebruiken in combinatie met dit concept. Momenteel is de scope van opleidingen nog zéér beperkt en zelfs dan bestaan er al regels die te algemeen zijn een duidelijk beeld te kunnen scheppen van wat er precies mis zou kunnen zijn. In een poging om de theorie nog algemener te maken zodat nog meer opleidingen beschreven kunnen worden, bleken de regels zo algemeen te worden dat een beschrijving in natuurlijke taal niet langer nuttig was. Dus waarschijnlijk zal er een afweging moeten gemaakt worden tussen een algemene theorie die zoveel mogelijk opleidingen kan beschrijven of duidelijke uitleg kunnen bieden ten koste van de uitdrukingskracht van de theorie.

Amilhastre

De derde en laatste methode zoekt naar alle oplossingen die mogelijk zijn om terug een selectie te bekomen die satisfieerbaar is. Hiervoor moet er eerst een automaat gebouwd worden. De bouw hiervan is zeer kostelijk en neemt enige tijd in beslag, maar hoeft slechts éénmaal uitgevoerd te worden. Dit is niet onverwacht, in zijn paper vermeldt Amilhastre ook dat de kost van deze constructie de moeite loont aangezien het gebruik van de automaat de complexiteit van de bewerkingen tijdens het selectieproces aanzienlijk verlaagt. De resultaten voor dit probleem zijn hier getuige van, de tijdsduur van de constructie zijn acceptabel voor een éénmalige offline compilatie. En de reactietijd van de inferentie methoden gedurende het selectieproces zijn ook zeer snel. /*INSERT RESULTS*/ Wellicht de belangrijkste opmerking is dat de bouw van de automaat mogelijk wordt gemaakt door gebruik te maken van inferentie technieken uit IDP. De automaat is een compacte representatie van alle oplossingen, om deze te kunnen opstellen moeten alle oplossingen gegenereerd worden. Dit kan in IDP simpelweg gedaan worden door model expansie uit te voeren en de resultaten op te sommen.

Chapter 4

Conclusion

Proof of concept De eerste onderzoeksvraag die gesteld werd in dit onderzoek was de vraag of het mogelijk was om met behulp van IDP een theorie te ontwikkelen voor het ISP probleem en belangrijker nog of we efficiënt inferentie kunnen doen met deze theorie. De voordelen van de uitdrukkingskracht van $FO(\cdot)$ zijn al vaker aangehaald ? ? in het verleden. En voor het huidige probleem houden deze beweringen opnieuw stand, sinds ik met behulp van $FO(\cdot)$ erin geslaagd ben om een theorie te ontwikkelen waarin de regels van het ISP vervat zitten. Deze theorie is zelfs in staat om meerdere opleidingen binnen het domein van informatica en computerwetenschappen te beschrijven. In IDP en het concept van kennis representatie is er een strikte scheiding tussen kennis en inferentie. De theorie wordt slechts éénmaal beschreven en deze laten toe om meerdere vragen te beantwoorden d.m.v. verschillende vormen van inferentie. Dit is enkel nuttig als deze inferentie technieken efficiënt en snel werken, en in dit geval snel genoeg om te kunnen voldoen aan de eisen van een interactief configuratie probleem. Het testen van deze technieken werd gedaan met behulp van de zelf ontwikkelde front-end applicatie. Voor alle gebruikte technieken bleken de reactiesnelheden ruim onder hun grenzen te liggen. Zelfs voor de meest complexe bewerkingen (bv. minimalisatie) boekt het IDP systeem goede resultaten. Dus met dit proof of concept wordt opnieuw de kracht van $FO(\cdot)$ getoond samen met de prestaties van de inferentiemethoden binnen het paradigma van kennis representatie.

Conflict explanation Meerdere onderzoeken hebben de voordelen van declaratieve methoden Gelle and Weigel (1996) en specifiek die van KR-systemen ? Denecker and Vennekens (2008) ? ? ten opzichte van imperatieve strategieën al aangehaald. Dit betekent niet dat een declaratieve aanpak op alle vlakken beter is, één voorbeeld daarvan is het domein van conflict explanation. In deze paper heb ik een aantal interessante technieken geïmplementeerd en de resultaten ervan beschreven. Twee hiervan, de unsatstructure en reified constraints maken enkel gebruik van de tools beschikbaar in IDP en één methode combineert technieken binnen IDP met die van ?. De unsatstructure werkt zoals verwacht, het is een ideale tool voor het opsporen van de oorzaak van niet-satisfieerbaarheid. Toegevoegd hieraan is de techniek van reified constraints, waarmee verklaringen in natuurlijke taal kunnen gegeven worden voor

regels die verbroken zijn. Hoewel deze aanpak gelimiteerd is in de verklaringen die het kan geven is ze daarbij ook nog eens sterk afhankelijk van de theorie. Gebaseerd op de in dit onderzoek gebruikte theorie, blijkt de uitleg over het algemeen voldoende duidelijk om de gebruiker te kunnen helpen in eventuele conflict situaties. Het grootste voordeel van reified constraints is uiteraard de lage implementatiekost. De laatste techniek tenslotte, gebruikt een automaat voor het berekenen van minimale oplossingen om de selectie terug satisfieerbaar te maken. De manier waarop deze automaat wordt gebouwt is zéér interessant omdat het gebruik maakt van model expansie, een inferentie techniek die ingebouwd zit in IDP. De resultaten van deze automaat zijn heel positief en het feit dat ze kan gebouwd worden met behulp van IDP biedt mogelijkheden voor de toekomst.

Chapter 5

Related Work

Separating Knowledge from Computation: An FO(\cdot) Knowledge Base System and its Model Expansion Inference ? Binnen het domein van declaratief programmeren, ziet men vaak de trend dat zo'n systeem een logische taal implementeert samen met een specifieke vorm van inferentie. Recent heeft men het paradigma Knowledge Base System of KBS voorgesteld met het idee dat kennis over een probleem niet gekoppeld hoeft te zijn aan één bepaalde vorm van inferentie. In plaats daarvan stelt men voor de kennis apart uit te drukken in een declaratieve taal waarop men dan verschillende vormen van inferentie kan toepassen. In deze paper wordt zo een systeem voorgesteld namelijk IDP. Het laat toe kennis over een bepaald probleem neer te schrijven in FO(\cdot), oftewel eerste orde logica met een aantal uitbreidingen zoals types, partiële functies, aggregaten en inductieve definities. Om mezelf bekend te maken met IDP en het paradigma van kennis representatie, is dit de paper bij uitstek. Het beschrijft het oorspronkelijke doel van IDP en motiveert de keuzes die zijn gemaakt bij het ontwerpen ervan. De paper begint met een korte samenvatting van eerste orde logica, aggregaten en inductieve regels en vervolgens wordt uitgelegd hoe men deze regels uitdrukt in de syntax van het IDP systeem. De verschillende inferentie technieken die ondersteund zijn zoals model expansion, optimization, deduction, model checking enzovoort worden toegelicht. Vervolgens geeft de paper een blik achter de schermen en beschrijft het de moeilijkheden van inferentie en de oplossingen die hiervoor voorzien zijn. Dit is een intro voor iedereen die zich wil verdiepen in IDP, het geeft je de wat, waarom en hoe die je nodig hebt om eraan te beginnen. De syntax in de paper is echter ondertussen niet meer up-to-date.

Interactive Configurations and the KB-Paradigm ? Deze paper gaat na of interactieve configuratie problemen kunnen opgelost worden m.b.v. declaratieve methodes en meer specifiek een KR-systeem. Algemeen bekend is dat dit soort problemen enorm complex zijn wil men ze programmeren in een imperatieve programmeertaal ?. De reden hiervoor is dat regels elkaar beïnvloeden en een simpele wijziging van ook maar één regel ervoor kan zorgen dat andere regels niet meer kloppen. De regels zijn ook vaak lang, onoverzichtelijk en slecht leesbaar, problemen die volgens de paper op te lossen zijn met IDP. Gebruik makend van FO(\cdot) en de

syntax van het IDP systeem gaat men proberen de constraints van zo 'n probleem te beschrijven en er vervolgens over redeneren door er verschillende vormen van inferentie op toe te passen. Specifiek wordt er gekeken naar de complexiteit van een aantal verschillende vormen van inferentie die IDP aanbiedt. Geconcludeerd wordt dat ondanks de limitaties van een KR-systeem (grote structuren kunnen tijd van het berekenen opblazen) stelt men toch vast dat de aanpak voldoende goede resultaten boekt. Men kan dit zien als een proof of concept waarbij een echt configuratie probleem uit de bedrijfswereld gebruikt wordt als voorbeeld. Een ISP selecteren is in essentie een interactief configuratie probleem, dus de resultaten die voortvloeien uit deze paper kunnen als leidraad gezien worden voor deze thesis.

A Logical Framework for Configuration Software ? In deze paper neemt de auteur opnieuw de klasse van configuratie problemen onder de loop. In deze verzameling van problemen staat software de gebruiker bij in het maken van bepaalde keuzes volgens een aantal regels. Het schrijven van software hiervoor met imperatieve programmeertalen is vaak een lastige opgave, het is een naïeve manier van opstellen waarbij elke keuze een if-else regel vereist en de kleinste verandering ervoor kan zorgen dat de regels niet meer kloppen. Dit is voor programmeurs vaak een nachtmerrie als de regels van een systeem geregeld wijzigen (bv. Tax on web). De auteur stelt een declaratieve aanpak voor en specifiek met het oog gericht op het model van kennis representatie. Het doel is om bestaande problemen efficiënter te kunnen oplossen. Als voorbeeld heeft men gekozen voor het probleem waar een student wordt gevraagd zijn opleiding samen te stellen volgens de regels van de universiteit. De regels kunnen gemakkelijk beschreven worden in IDP, dat gebruik maakt van een uitgebreide versie van FO(.). De expressiviteit ervan wordt gezien als een groot pluspunt en maakt het heel gemakkelijk voor de programmeur om regels elegant en leesbaar te beschrijven. Er worden verscheidene vormen van inferentie toegepast in de paper waaronder model expansion, propagation, model checking en conflict explanation. Opvallend is dat al deze technieken werken met dezelfde domein logica, één van de kern ideeën en grote voordelen van het KR-systeem. Er wordt beschreven hoe elke techniek te werk gaat en uiteindelijk toont de paper aan dat voor het gekozen probleem de inferentie voldoende efficiënt en in polynomiale tijd verloopt. Hierbij komt ook nog dat de elegantie van een declaratieve aanpak, de grote expressiviteit van FO(.) en de efficiëntie van een kennis representatie systeem vele voordelen bieden in tegenstelling tot imperatieve technieken. De technieken die de paper beschrijft voor het oplossen van het gekozen configuratie probleem kunnen als basis dienen voor mijn verder werk en zijn uitermate geschikt om op verder te bouwen. De bedoeling is om technieken toe te voegen zoals minimization en de theorie verder uit te breiden met kennis over het lessenrooster.

Predicate Logic as a Modelling Language: The IDP System ? Recente doorbraken in automated reasoning technieken zoals SAT-solving and constraint programming zijn de aanleiding tot een groeiende interesse in logica als programmeertaal. Ondertussen zijn er al verscheidene declaratieve programmeertalen verschenen. Hier

stelt men IDP voor als kennis representatie systeem, waar domein logica volledig onafhankelijk is van de inferentie die men erop wil toepassen. Imperatieve talen laten toe een specifiek algoritme te schrijven dat hoge prestaties kan leveren. Hier tegenover staat dat kennis over het probleem verweven zit in de code, wat onderhoud enorm moeilijk of soms zelfs onmogelijk maakt. Vaak kunnen ze ook maar één taak uitvoeren, in tegenstelling tot IDP waar verschillende inferentie taken dezelfde domein logica gebruiken. De syntax in IDP is makkelijk te begrijpen en leunt dicht aan bij natuurlijke taal, iets wat bij imperatieve programmeertalen vaak niet het geval is. En tenslotte hoeft de programmeur in IDP zich geen zorgen te maken over hoe een probleem moet opgelost worden. Hij moet enkel de domein logica beschrijven en IDP doet de rest. Er zijn ook limitaties, imperatieve oplossingen zijn vaak sneller dan een algemene declaratieve aanpak. Toch worden de prestaties van IDP zeker voldoende geacht. De rekentijd is voldoende kort dat het kan gebruikt worden in een interactieve omgeving waarbij de gebruiker in minder dan enkele seconden een antwoord verwacht. Dit is een van de belangrijkste resultaten van de paper aangezien het aantoonde dat IDP wel degelijk in staat is dit en andere gelijkaardige problemen met voldoende efficiëntie op te lossen. Naast de motivatie van de paper worden alle belangrijke aspecten van IDP uitgelegd. Zo wordt beschreven hoe de FO(\cdot) in mekaar zit, waarna de paper toont hoe men dit in de syntax van IDP zelfs kan schrijven. Er wordt ook een motivatie gegeven waarom men de keuze van FO(\cdot) heeft gemaakt, belangrijk hier zijn vooral de expressiviteit, begrijpbaarheid en herbruikbaarheid. Ook wordt uitgelegd hoe IDP under the hood werkt met technieken zoals ground-and-bound, symmetry exploitation en functional dependency detection.

Consistency Restoration and Explanations in Dynamic CSP's: Application to Configuration ? In de klasse van interactieve configuratie problemen is het belangrijk dat de technieken snel en efficiënt werken. Bepaalde functionaliteiten zoals het voorzien van uitleg bij een foute keuze van de gebruiker en herstellen van consistentie zijn echter nog altijd moeilijk op te lossen en nemen teveel tijd in beslag om ze interactief te kunnen gebruiken. In deze paper proberen ze een oplossing te bieden voor deze problemen. Het voorstel is om een automaat te bouwen die alle verschillende mogelijke selecties voorstelt. De voorwaarde is wel dat deze automaat vooraf off-line wordt opgesteld, dit hoeft maar één maal te gebeuren. Gebruik makend van deze automaat toont de paper aan dat de voorheen vermelde technieken op een efficiënte en snelle manier kunnen berekend worden. Om hun worden kracht bij te zetten hebben ze een configuratie probleem van Renault voor het configureren van een auto gebruikt om hun model te testen. De resultaten ervan zijn trouwens zeer indrukwekkend. IDP voorziet momenteel wel een techniek voor het opsporen van conflicten. Maar de technieken voorgesteld in de paper zijn veelbelovend en het is zeker interessant om eens te kijken of we deze toch niet in IDP kunnen integreren.

Lowering the learning curve for declarative programming: a Python API for the IDP system ? Nog te vaak zijn programmeurs terughoudend als het aankomt op declaratief programmeren vanwege de leercurve die ermee gepaard gaat.

Het doel van de auteur is om de moeilijkheidsgraad te verlagen in de hoop dat het de stap naar declaratieve systemen vergemakkelijkt. De auteur probeert dit te bereiken door een IDP KR-systeem te integreren in een Python API. Python is één van de meest gebruikte programmeertalen, en dankzij de API kan iedereen vertrouwd met Python gebruik maken van het IDP KR-systeem zonder de syntax te moeten leren. Doch zijn declaratieve systemen inherent verschillend van imperatieve systemen, en iedereen die geen opleiding omtrent declaratieve systemen heeft genoten zal dit dan toch nog moeten inhalen. Bij studenten Computerwetenschappen is dit echter wel het geval aangezien het standaard onderdeel is van de opleiding. En de auteur hoopt met de API dat studenten sneller de stap zullen maken naar IDP, door ze de mogelijkheid te geven om dit te doen in de vaak vertrouwde Python omgeving. De API is zeker een goede overbrugging, maar ik ben ondertussen bekend met de syntax van IDP. En de efficiëntie van de syntax en de goede leesbaarheid ervan hebben mij toch doen kiezen om geen gebruik te maken van de API.

Generating Corrective Explanations for Interactive Constraint Satisfaction ? In het veld van interactieve configuratie problemen is het zo dat als de gebruiker een ongeldige combinatie van variabelen selecteert we op zoek gaan naar de oorzaak hiervan. Anders gezegd gaan we een verklaring zoeken voor het probleem. Dit is natuurlijk handig voor de gebruiker om te achterhalen welke selectie hij moet aanpassen om het conflict op te lossen. Maar nog handiger zou zijn dat niet enkel de oorzaak wordt getoond, maar ook een (of meerdere) mogelijke oplossing(en), m.a.w. een selectie die zo veel mogelijk dezelfde is als die van de gebruiker, maar die wel tot een mogelijk juiste oplossing kan leiden. De paper stelt **CORRECTIVEEXP** voor, een systematisch algoritme voor het zoeken van minimal corrective explanations. Een vergelijkende studie met reeds bestaande technieken op verscheidene grote problemen toont aan dat het algoritme zeer goede resultaten boekt. Dit is essentieel aangezien dat interactieve configuratie problemen een snelle reactietijd vereisen.

Chapter 6

Future Work

Voor een klein domein van opleidingen is het gelukt om een theorie te ontwerpen met behulp van $FO(\cdot)$. Natuurlijk is het aantal opleidingen binnen de K.U. Leuven aanzienlijk groter. Dus achterhalen of het mogelijk is om een theorie te ontwikkelen die om kan gaan met alle opleidingen, is werk voor de toekomst. Maar ik zou niet enkel zoeken naar een antwoord op dit probleem maar ook naar alternatieven zoeken, bijvoorbeeld één theorie per type opleiding. Bachelor opleidingen hebben vaak meer gemeen met andere bachelor opleidingen, dan met bijvoorbeeld master studies. Het zou zeker interessant zijn om te kijken wat de mogelijkheden hieromtrent zijn en voor- en nadelen die hiermee gepaard gaan.

De prestaties van de inferentie technieken van IDP voor het ISP probleem zijn positief en voldoen aan de vereisten van een IC probleem. De front-end is in staat om alle functionaliteit van de bestaande web applicatie aan te bieden met toevoeging van nieuwe functionaliteiten. Deze zijn *bijna* allemaal mogelijk gemaakt door de technieken die in IDP te vinden zijn. Deze succesvolle steekproef op een klein subdomein van opleidingen leidt mij tot de conclusie dat er meer mogelijk is. Hopelijk zijn deze resultaten een motivatie voor anderen om verder uit te zoeken wat de mogelijkheden zijn van IDP met betrekkingen tot het ISP probleem en andere (als niet alle) opleidingen aan de K.U. Leuven en misschien zelfs andere onderwijsinstellingen.

Het IDP systeem alleen volstond echter niet om alle functionaliteiten van de front-end te kunnen ondersteunen. Door gebruik te maken van de automaat die beschreven staat in het werk van Amilhastre et al. is het gelukt om de gewenste vorm van conflict explanation te implementeren. Opvallend hierbij is dat de bouw hiervan kan gebeuren op basis van model expansie in IDP. Ondertussen wordt er hard gewerkt aan een nieuwe versie van het IDP systeem. Gezien de voordelen van het gebruik van zo'n automaat zou het zeer interessant zijn om te zien of het werk van Amilhastre niet geïntegreerd kan worden in een toekomstige versie van het systeem.

Bibliography

Marc Denecker and Joost Vennekens. Building a knowledge base system for an integration of logic programming and classical logic. In Logic Programming, pages 71–76. Springer, 2008.

Esther Gelle and Rainer Weigel. Interactive configuration using constraint satisfaction techniques. In PACT-96, pages 37–44, 1996.

Joost Vennekens. Lowering the learning curve for declarative programming: a python api for the idp system. arXiv preprint arXiv:1511.00916, 2015.

Fiche masterproef

Student: Herbert Gorissen

Titel: The ISP Course Selection Puzzle

Nederlandse titel: The ISP Course Selection Puzzle

UDC: 621.3

Korte inhoud:

Here comes a very short abstract, containing no more than 500 words. \LaTeX commands can be used here. Blank lines (or the command \backslash par) are not allowed!

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: elektrotechniek, optie Elektronica en geïntegreerde schakelingen

Promotor: Prof. Gerda Janssens

Assessor:

Begeleider: Matthias van der Hallen