KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

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

Academiejaar 2015 – 2016

Preface

Herbert Gorissen

Contents

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 genoemd 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 in het IDP systeem.

List of Figures and Tables

List of Figures

List of Tables

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 van het type problemen waar men voor een reeks variabelen een geldige waarde uit het domein dient toe te kennen. 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 valt in de klasse van NP-Compleet problemen 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 men een toekenning van domeinwaarden voor de variabelen wil zoeken waarvoor geldt dat aan alle constraints voldaan is. Bij IC problemen wil men exact hetzelfde bekomen maar waar men voorheen recursief ging zoeken naar zo'n oplossing is het nu een gebruiker die handmatig stap per stap een geldige toekenning gaat proberen selecteren. Om de gebruiker hierin bij te staan is er nood aan software, software die de gebruiker zo goed mogelijk bijstaat gedurende dit hele proces. Men weet uit ervaring dat het schrijven van software voor

dit soort problemen geen gemakkelijke opgave is. Regels ontwikkelen voor kleine problemen zoals het n-queen probleem zijn intuïtief en gemakkelijk. Maar naarmate de problemen groter en complexer worden zal het als maar moeilijker worden om dit in software om te zetten. Verder 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. Daarboven 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. Daarom is men op zoek gegaan naar alternatieven en het antwoord kwam uit de declaratieve hoek. Met de opkomst van declaratieve programmeertalen, vooruitgang in automated reasoning en als maar toenemende rekenkracht van computersystemen is de populariteit van declaratieve toepassingen alleen maar gegroeid. In declaratieve toepassingen kan men de regels van een probleem beschrijven d.m.v. logische constraints. Wellicht het grootste voordeel van declaratieve methoden in tegenstelling tot imperatieve is dat de regels overzichtelijker en duidelijker zijn. Onderhoud van deze constraints is veel makkelijker, als er een regel wijzigt dan hoeft ook enkel deze constraint aangepast te worden. En hoewel men met een imperatieve aanpak optimalere algoritmen kan ontwikkelen, heeft men kunnen laten zien dat declaratieve methoden ook goede prestaties kunnen neerzetten (**?**).

1.0.3 Knowledge Representation: Het IDP Systeem

De interesse in declaratieve talen is de laatste decennia toegenomen en als resultaat zijn er een hele resem talen verschenen (Prolog, Ant, Lisp, ...). Wat opvalt is dat elke taal vaak samenhangt met één enkele specifieke vorm van inferentie. En toch is ondanks de verschillende vormen van inferentie de kennis over het domein hetzelfde. Enkele jaren geleden heeft men het concept van Knowledge Representation (Denecker and Vennekens, 2008) voorgesteld. In het KR-paradigma stelt men dat ongeacht de inferentie die men wil toepassen, men hiervoor telkens dezelfde kennis kan hergebruiken. Deze kennis is niets meer dan een verzameling van informatie, maar met deze informatie kan men meerdere vormen van inferentie toepassen. /*UITLEG OVER IDP ZELF*/

1.1 Probleemstelling

1.1.1 Individueel Studieprogramma

Een ISP samentstellen valt onder deze 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. Maar de student kan niet zomaar elk opleidingsonderdeel selecteren, er zijn een heleboel regels waaraan de selectie moet voldoen. Deze regels Naast al deze regels moet er ook nog rekening gehouden worden met het lessenrooster. Het is mogelijk dat lesmomenten voor

verschillende opleidingsonderdelen overlappen met elkaar. Het probleem is echter dat de student hier niet van op de hoogte wordt gebracht noch een overzicht heeft.

1.2 Doel

In deze thesis willen we onderzoeken of de regels van het ISP efficiënt kunnen beschreven worden in FO(·). Met de bedoeling dat voor eender welke opleiding binnen de K.U. Leuven deze set van regels een geldige opleiding kunnen beschrijven.

Vervolgens willen we kijken of we met deze theorie verschillende vormen van inferentie kunnen doen en hoe efficient deze gebeuren. Met behulp van een zelf ontworpen front-end (incl. grafische interface) willen we nagaan of de gebruiker een geldig ISP kan selecteren, kan laten samenstellen enz. En of al deze functionaliteiten in real-time kunnen uitgevoerd worden met een reactietijd van maximaal enkele seconden.

Momenteel wordt het lessenrooster niet in acht genomen tijdens het selectieproces. Dit wil zeggen dat de student dus ook geen idee heeft of de door hem/haar geselecteerde opleidingsonderdelen mogelijk lesmomenten bevatten die overlappen. Hier is het de bedoeling om de gebruiker een overzicht te geven van het lessenrooster voor elke stap in het selectieproces. En daarboven het beste lessenrooster te laten genereren waarbij er zo weinig mogelijk overlappende lessen zijn.

In het geval van een ongeldige selectie biedt IDP een aantal mogelijkheden om aan de gebruiker duidelijk te maken wat er juist mis is gegaan. De *unsatcore* zoekt naar de minimale set van regels die ongeldig zijn voor de huidige selectie. Een andere optie is de *unsatstructure* die de kleinste set van variabelen wordt gezocht waarbij als men de geselecteerde waarden voor deze variabelen ongedaan maakt, de selectie niet langer ongeldig is. Maar om een duidelijk verklaring te verkrijgen waarom de selectie fout is, in een formaat dat elke persoon kan begrijpen bestaat er in IDP momenteel nog niets. Daarom willen we onderzoeken of we dit kunnen realiseren.

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 uitdrukkingskracht, 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 iedereen opleiding in te lezen.

Chapter 2

Implementation

2.1 Knowledge base constructie

2.1.1 Theorie van het ISP

Geen opleiding aan de KU Leuven is dezelfde, de opleidingsonderdelen verschillen net zoals de vakgroepen waar ze deel van uitmaken. Voor sommige vakgroepen ben je verplicht alle onderdelen op te nemen, terwijl je voor andere en minimaal (of zelfs maximaal) aantal studiepunten ervan moet opnemen. Kortom de regels binnen éénzelfde opleiding zijn niet zo moeilijk om te beschrijven, maar om dit te veralgemenen en een theorie te creëren die voor alle opleidingen een geldig ISP kan beschrijven dat is een andere paar mouwen. Voor het gemak van de lezer noemen we opleidingsonderdelen vanaf nu vakken.

Vakken

Een vak is een simpel type dat altijd dezelfde eigenschappen heeft vertoond. Het heeft een unieke vakcode met daaraan een naam gekoppeld. Het telt een aantal studiepunten dat de werklast beschrijft. Een opleiding kan kan bestaan uit één of meerdere fases, en voor elk vak is er bepaald in welke fase(s) van de opleiding je het kan volgen. let wel je kan een vak per opleiding maar één keer volgen. En tenslotte valt elk vak oftwel in het eerste, tweede of beide semesters (in dit laatste geval noemt men het een jaarvak).

Vakgroeptypes

Zo valt op dat we telkens dezelfde **types** van *vakgroepen* tegenkomen. 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 eeen 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 1 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 laatse 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 verbrendend pakket.

2.1.2 Theorie van het lessenrooster

2.2 Front-End

Om een zo goed mogelijke gebruikservaring aan te kunnen bieden hebben we een Front-end applicatie ontwikkeld voorzien van een Grafische Interface. Deze communiceert under the hood met de IDP knowledge base. Alle selecties gemaakt door de gebruiker worden vertaald naar FO(·) en doorgespeeld aan IDP. Vervolgens voert IDP de gewenste vorm van inferntie uit en geeft het resultaat hiervan terug aan de Front-end die dan verwerkt en aan de gebruiker laat zien via de GUI.

2.2.1 Grafische Gebruikersinterface

2.2.2 Communicatie tussen Front- en Back-end

Niet alleen zijn 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). En hoewel deze API zeker de kloof tussen beide programmeertalen kleiner maakt, schiet ze op sommige vlakken toch te kort. Zo laat het bijvoorbeeld niet toe om aggregaties te te beschrijven. Daarom heb ik gekozen om ze niet te gebruiken en in plaats daarvan zelf de communicatie tussen beide te regelen. De theorie en vocabulair worden uitgelezen uit een tekstbestand en de structuur kan automatisch gegenereerd worden door de Vak en Vakgroep klassen in Python zelf. Een parser klasse steekt alles samen in een string en geeft dit door aan de IDP solver. Het resultaat wordt dan gefilterd (de nieuwe waarden worden eruit gehaald) en de veranderingen worden verwerkt.

2.3 Features

2.3.1 Propagatie

Om het ISP samen te stellen moeten volgende, nog onbekende predicaten ingevuld worden door de gebruiker: Geselecteerd(Vak,Fase) en NietGeintereseerd(Vak). Bij elke stap (nieuwe selectie die de gebruiker maakt) wordt indien die selectie satisfieerbaar is, eventuele propagaties ook geselecteerd.

2.3.2 Model Expansie

Als de gebruiker een aantal selecties heeft gedaan en voor de rest geen specifieke eisen meer heeft, kan hij vragen aan het systeem om het ISP verder te laten genereren.

2.3.3 Minimizatie

Ook is het mogelijk om het systeem het beste ISP te laten genereren volgens bepaalde regels. Momenteel is het mogelijk om een ISP te zoeken dat zo weinig mogelijk werklast bevat (zo min mogelijk studiepunten). Of om de werklast zo goed mogelijk te verdelen over de beide semesters. Daarnaast is het ook mogelijk om de vakken zo te selecteren dat hun lesmomenten zo weinig mogelijk overlappen.

2.3.4 Selectieproces

Elke keuze die de gebruiker maakt moet zorgvuldig afgehandeld worden willen we dat de selectie altijd satisfieerbaar is. De voorwaarde is dat als de gebruiker een keuze maakt die ervoor zorgt dat de theorie onsatisfieerbaar wordt, hij hier meteen op de hoogte van wordt gebracht en dit dient aanpassen.

De werking is als volgt, bij elke nieuwe keuze wordt eerst gekeken of deze nieuwe selectie samen met alle voorgaande keuzes een satisfieerbare structuur vormen. Zo niet dan wordt de unsatstructure gezocht. Anders worden die eventueel nieuwe propagaties berekend. De nieuwe structuur wordt vergeleken met de voorgaande, om na te gaan of eventuele propagaties niet meer van kracht zijn. Als dit het geval is dan krijgt de gebruiker de keuze om ze te behouden. Dit hij dit dan wordt dit gezien als een selectie en dus begint het hele proces opnieuw.

2.3.5 Ongedaan Maken

Het ongedaan kunnen maken van selecties is een van de aspecten waar meerdere strategiëen mogelijk zijn. Belangrijk op weten is dat er een duidelijk onderscheid gemaakt moet worden tussen de keuzes gemaakt door de gebruiker en de eventuele propagaties die hieruit volgen. Deze moeten dan ook voor elke tussenstap keurig bijgehouden worden in wat we vanaf nu een actie zullen noemen. Het is eveneens belangrijk om de originele waarden bij te houden, zodanig dat als men een actie ongedaan maakt de toestand terug kan gezet worden naar hoe ze voorheen was. Het voorgaande is allemaal redelijk vanzelfsprekend, maar de moeilijkheid zit hem in de verschillende mogelijke strategieën.

Algorithm 1: Selectieproces

```
1 function Selectieproces (U_i) ;
  Input  : De nieuwe keuze van de gebruiker U_i
  Output : U_i, nieuwe propagaties P_i
2 U ← U_1 ∪ ... ∪U_i;
3 if Sat(U) then
4 |  P_i ← Propagate(U_1);
5 else
6 |  V ← Unsat(U);
7 end
```

Strategie 1 Een mogelijkheid is om simpelweg een actie te zien zoals een overgang van één consistente toestand naar een nieuwe consistente toestand. Dus elke actie weet hoe de selectie eruit zag voordat ze uitgevoerd was, en hoe de selectie eruit zag na afloop van de actie. Een actie ongedaan maken houdt dan simpelweg in dat je de nieuwe toestand terug vervangt door de oude, een beetje als een rollback operatie. De voorwaarde hier is dat enkel de laatste actie ongedaan gemaakt kan worden, alvorens mijn de voorlaatste ongedaan wil maken.

Strategie 2 Dat brengt ons bij het volgende, bij het ongedaan maken van een actie draaien we als het ware de tijd terug en wordt de huidige toestand vervangen door de voorgaande. Dit wil zeggen dat ook de propagaties uit de actie verdwijnen, en misschien wil de gebruiker dit niet. In plaats daarvan willen we bij het ongedaan maken van een operatie de gebruiker laten kiezen of de eventuele propagaties mogen geselecteerd blijven of dat ze ook ongedaan gemaakt mogen worden. Hier komen we het volgende probleem tegen. Stel dat het vak A geselecteerd is en hieruit volgt dat vak B ook gevolgd moet worden. In de volgende actie selecteert de gebruiker dat hij niet geïntereseerd is in A en hieruit volgt dat B niet gevolgd mag worden. De gebruiker komt terug op zijn beslissing en wil de laatste actie ongedaan maken. Dit wil zeggen dat de keuze van vak A van niet geïntereseerd terug gezet wordt naar wel geselecteerd. Hieruit volgt dat de propagatie over vak B niet meer van kracht is, en de gebruiker dus de keuze krijgt om deze ongedaan te maken of niet. Stel dat hij ervoor kiest de propagatie te behouden dan wordt het probleem onsatisfieerbaar. Dit is niet het enigste probleem dat zich voordoet met deze strategie. Iedere actie beschrijft hoe de toestand voor en na de actie. Als de gebruiker een actie ongedaan maakt maar ervoor kiest om de propagaties uit deze actie toch te behouden, dan wil dit dus zeggen dat het resultaat van de voorgaande actie niet meer klopt met de huidige situatie. In feite maken we dus niet heel de actie ongedaan wat het heel moeilijk maakt om een duidelijk overzicht te bieden en consistentie te garanderen. Dat brengt ons bij het volgende punt. We willen actie Ai ongedaan maken, de actie bevat Ui de keuze van de gebruiker en Pi de propagaties die eruit volgden. De gebruiker kiest ervoor om p1 ∈ Pi te behouden. Ai wordt ongedaan gemaakt, maar nu wordt er een actie A'i toegevoegd waarin U'i =p1 de nieuwe selectie van de gebruiker is en P'i de

eventuele propagaties die hieruit volgen.

Strategie 3 Tenslotte is er nog een geheel andere optie, waarbij niet meer uit wordt gegaan van acties die men kan ongedaan maken. Een vak kan geselecteerd zijn in een fase, het kan zijn dat de gebruiker niet geïnteresseerd is in een vak of het kan zijn dat de gebruiker nog geen keuze heeft gemaakt. Het idee van iets ongedaan maken bij deze strategie is het volgende: de gebruiker heeft geselecteerd dat hij/zij vak A wil volgen tijdens fase 1 van de opleiding. Later zegt de gebruiker dat hij zij dit niet meer wel, maar zegt verder specifiek niets over het vak. We hebben de volgende verzamelingen: Ci de vakken waar de gebruiker specifiek een keuze over heeft gemaakt, Pi de propagaties die volgen uit Ci en Ui de vakken waarover nog geen uitspraak is gedaan. Stel vak V $\in$ Cn-1 maar de gebruiker maakt de selectie ongedaan en V wordt toegevoegd aan Un. Dus Cn = Cn-1 - V. dan gaan we kijken wat Pn en Un zijn, en vervolgens gaan we zien wat de doorsnede is van Pn-1 en Un, is deze verzameling niet leeg, dan wil dit zeggen dat er propagaties zijn die niet langer bestaan, en de gebruiker moet kiezen of hij/zij ze wil behouden. Zo ja worden ze toegevoegd aan Cn+1 en begint het hele process opnieuw.

2.4 Conflict Explanation

Het is mogelijk dat de gebruiker een verkeerde keuze maakt waardoor de theorie nietmeer 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 niet meer waar kunnen worden gemaakt.

A $\wedge$ B
$\neg$ A

In het voorbeeld, zegt de theorie ons dat A en B beide waar moeten zijn. We zeggen dat A niet waar is. Door deze invulling kan de regel nooit waar worden ongeachte de waarde van B. De theorie is dus onsatisfieerbaar.

Conflict explanation wil zeggen een verklaring zoeken waarom een bepaalde selectie door de gebruiker dit veroorzaakt.

IDP gebruikt momenteel twee technieken om oorzaken van onsatisfieerbaarheid op te sporen. De unsatstructure spoort de set van variabelen op die het probleem veroorzaken. Vervolgens is er de unsatcore, die zoekt naar de kleinste set van regels uit de theorie die niet waar gemaakt kunnen worden.

Unsatstructure

De unsatstructure is een efficiënte tool om aan de gebruiker te kunnen meedelen, welke van de voormalige selecties problemen veroorzaken. In ons voorbeeld van het ISP kan de gebruiker enkel de keuze maken om een dag te volgen in een bepaalde fase (te kiezen uit de lijst van mogelijke waarden) of kan hij/zij aangeven om een vak niet te volgen. Bij een slechte selectie zal de unsatstructure de vakken oplijsten samen met de selectie die de gebruiker ervoor gemaakt heeft, en dit enkel voor de vakken

waarvoor de selectie bijdragen tot de onsatisfieerbaarheid. De gebruiker krijgt hier dan de mogelijkheid om voor deze vakken de voorheen gemaakte keuze ongedaan te maken of te veranderen.

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 kan niet weten wat er moet veranderen. Herinner de unsatcore zoekt achter de kleinste set van regels uit de theorie die niet meer waar gemaakt kunnen worden. En hoewel dit hetgene is dat we willen weten, is er toch nog een probleem. De IDP syntax is voor de doorsnee programmeur goed te lezen, maar voor een doorsnee gebruiker die niets van programmeren afweet zal dit ongetwijfeld als chinees overkomen. Neem als voorbeeld onderstaande regel, elke student computerwetenschappen zal vrij snel kunnen achterhalen wat de regel inhoud. Maar personen zonder deze achtergrond zullen dit niet kunnen verstaan.

Listing 2.1: IDP Rule Example

```
∀vg[VakGroep] : IsType(vg,AVO) ⇒ GesAantalStupunVakGr(vg)
= sum{v[Vak], sp[Studiepunten],f[Fase] : InVakGroep(v,vg)
∧ Geselecteerd(v,f) ∧ AantalStudiepunten(v)=sp : sp }.
```

We proberen dit probleem deels te overbruggen d.m.v. reified constraints. Hierbij koppelen we de waarheidswaarde van een regel (constraint) uit de theorie aan een booleaanse variabele. Nemen we terug onze regel uit voorbeeld 1, daarvoor plaatsen we de Booleaanse variabele C toe gevolgd door een equivalentie. Wat dit wil zeggen is dat de waarheidswaarde voor en na de equivalentie dezelfde moet zijn. Dus als de regel uit voorbeeld 1 niet voldaan is, en dus 'false' is dan moet C ook 'false' zijn.

$C \Leftrightarrow A \wedge B$
$\neg A$

Op deze manier hoeven we enkel na te gaan of C true of false is om te weten te komen of de regel waar is of niet. Hiermee is de gebruiker natuurlijk nog niets wijzer geworden, dus wat is nu net het nut van dit soort constraints? Voor elke regel uit de theorie hebben we een uitleg in natuurlijke taal geschreven die beschrijft wat er mis is als de regel niet consistent is. Daarna moeten we simpelweg voor elke regel controleren wat de waarheidswaarde is van de booleaanse variabele voor de equivalentie. Is deze false, dan is de regel dus inconsistent en krijgt de gebruiker de uitleg te zien van deze regel in natuurlijke taal.

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 laat deze methode enkel toe om inconsistente regels op te sporen. Een regel is inconsistent als ze niet voldaan is gegeven de huidige selectie. Terwijl een regel die nooit waar gemaakt kan worden gegeven de huidige selectie onsatisfieerbaar is. En ten tweede is de uitleg vaak oppervlakkig en niet specifiek

genoeg om duidelijk te kunnen aanwijzen welke selectie juist verantwoordelijk is voor de inconsistentie. Desalniettemin geeft deze techniek de gebruiker redelijk goed zicht op het probleem en dragen ze toch bij tot het maken van juiste beslissingen. Een voordeel is dat zodra er geen inconsistente regels meer weergegeven zijn de gebruiker weet dat de gemaakte selectie consistent is met de theorie.

Chapter 3

Evaluation

De bedoeling van dit onderzoek is om de regels van het ISP te kunnen beschrijven in IDP, zodanig dat ze voor elke opleiding aan de KU Leuven een correct model kunnen genereren. 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 conflict explanation, waar bij een foute selectie de gebruiker een specifieke uitleg hoort te krijgen wat er mis is, waarom en hoe het opgelost kan worden. Dit alles in een zo natuurlijk mogelijke taal die de gebruiker gamakkelijk kan verstaan.

Theorie van het ISP Geen twee opleidingen zijn dezelfde, en toch is het belangrijk dat 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 structeren 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.

Het lessenrooster

Grafische Interface De GUI dient als prototype voor een mogelijke versie die de student kan gebruiken in de toekomst. Daarin moet het minstens de functionaliteit bieden van het huidig systeem, maar met een aantal verbeteringen. Eerst en vooral is het de bedoeling dat de student de verschillende inferentietechnieken van de IDP back-end kan aanroepen om hulp te bieden bij het

selectieproces. Het huidige systeem geeft enkel bij bevestiging terug of de selectie al dan niet correct is volgens de regels. De nieuwe GUI doet dit in real-time op het moment van de foutieve selectie en weergeeft de oorzaak van het probleem. Naast dit alles is het ook de bedoeling om in het nieuwe systeem het lessenrooster te betrekken. De student krijgt een weekoverzicht te zien van de lessen voor de vakken die hij/zij geselecteerd heeft. Zo kan er worden gecontroleerd of de lessen mogelijk overlappen. Daarbij is het ook mogelijk om IDP een ISP te laten samenstellen waarbij de lessen zo min mogelijk overlappen. De ontwikkeling van de GUI was zeer kostelijk, en heeft langer geduurd dan eerst gehoopt. De hoge configureerbaarheid van de elementen heeft tot gevolg dat het meer tijd kost om ze correct te implementeren. Hoewel de kv-language dit process vergemakkelijkt ligt de tijdsduur nog altijd hoger dan verwacht. Een van de belangrijkste criteria waar ik niet naar heb gekeken bij het kiezen van het framework is KIS (Keep It Simple). En dit zal ik zeker niet vergeten in de toekomst.

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 minimizatie opdrachten geven goede resultaten terug met een maximale rekentijd van ongeveer 10 seconden.

Conflict Explanation Als de gebruiker een keuze maakt waardoor de regels onsatisfieerbaar worden, dan is het de bedoeling dat de oorzaak hiervan opgespoord wordt en de gebruiker hierover wordt ingelicht. Via de unsatstructure van IDP krijgt de gebruiker een lijst te zien met vakken waarvoor hij/zij een keuze heeft gemaakt die de onsatisfieerbaarheid veroorzaken. Het is in deze lijst dat de gebruiker keuzes kan veranderen of ongedaan maken om det probleem op te lossen. 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 constraint bieden extra informatie en dienen als een helpende hand om de gebruiker door het selectieprocess te loodsen. Deze simpele generische manier om inconsistente regels op te sporen en te verklaren in natuurlijke taal heeft zeker zijn voordelen. De hoeveelheid werk vereist om deze functionaliteit te implementeren is zeer weinig. Maar de functionaliteit van deze techniek is echter beperkt. Het laat toe om inconsistente regels op te sporen in tegenstelling tot wat we zoeken namelijk de oorzaken van onsatisfieerbaarheid. Daarbij is

de uitleg die voorzien wordt bij een inconsistente regel mogelijk niet toereikend voor de gebruiker, zo kan er niet aangewezen worden voor welke instantie van de variabelen een bepaalde regel inconsistent is, en het concept dat de regel beschrijft is vaak heel algemeen waardoor de gebruiker er moeilijk iets uit kan afleiden. Ondanks deze minpunten is deze techniek zeker nuttig in de zin dat het de gebruiker in de juiste richting kan leiden tijdens het selectieproces, en dit voor een lage implementatiekost.

Amilhastre paper

Chapter 4

Conclusion

Chapter 5

Related Work

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 `\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