KU LEUVEN

FACULTEIT WETENSCHAPPEN

# Programmeren met Onzekerheid: een Case Study

**Sus VERWIMP**

Promotor: Prof. T. Schrijvers
Affiliatie *(facultatief)*

Begeleider: *A. Vandenbroucke (facultatief)*
Affiliatie *(facultatief)*

Proefschrift ingediend tot het behalen van de graad van Master of Science in Toegepaste Informatica

Academiejaar 2017-2018

# Voorwoord

# Korte Samenvatting

# Lijst van afkortingen en lijst van symbolen

# Inhoudsopgave

# Hoofdstuk 1

# Inleiding

De grote interesse in het domein onzekerheid in AI resulteerd in de ontwikkeling van verschillende programmeertalen. Deze talen worden probabilistische programmeertalen of PPL (Probabilistic Programming Language) genoemd. Een PPL heeft in grote lijnen 2 hoofdfuncties:

- Het modelleren van een wereld met onzekerheid
- Het redeneren/infereren van vragen over deze wereld

Voorheen was het zeer moeilijk als programmeur om werelden met onzekerheid te modelleren, laat staan het redeneren over deze werelden. Met de komst van PPLs is dit probleem al een stuk makkelijker geworden.

Elke PPL heeft zijn eigen manier van implementatie en wordt vaak geimplementeerd als extensie op een general-purpose programmeertaal. Dit zorgt er voor dat de PPL niet gelimiteerd is aan een kleine subset van de werelden die het kan modelleren. Veel van deze PPLs streven naar een balans tussen performantie en expressiviteit. Het is belangrijk voor een PPL om genoeg werelden te kunnen simuleren en op een aanneembare tijd te kunnen redeneren over vragen over deze wereld.

Omdat er zo veel PPLs beschikbaar zijn de laatste jaren is het niet altijd duidelijk welke voordelen of nadelen ze hebben ten opzichte van andere PPLs. Verschillende van deze PPLs zijn in recente artikels vergeleken met elkaar aan de hand van eigenschappen en concepten van de taal (Probabilistic (Logic) Programming Concepts, Luc De Raedt - Angelika Kimmig). Andere artikels vergelijken PPLs aan de hand van hun vorige iteratie of PPLs met hetzelfde programmeerparadigma (Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas, LUC DE RAEDT et al).

In deze thesis ga ben ik van plan PPLs met een verschillend programmeerparadigma te evalueren. ik ga deze PPLs evalueren ten opzichte van elkaar aan de hand van qualitatieve en quantitatieve criteria. Omdat het niet triviaal is om programmeertalen te vergelijken die totaal anders geimplementeerd zijn maak ik gebruik van een case study. Ik begin met het verzinnen van een onzekerheidsprobleem waarna ik uitleg geef waarom ik gekozen heb voor dit probleem. Daarna evalueer ik de implementatie van het probleem in de verschillende PPLs aan de hand van vooropgestelde criteria. Ten slotte volgt een

evaluatie van de PPLs ten opzichte van elkaar.

Onzekerheid in artificiële intelligentie is een van de invloedrijkste domeinen van artificiële intelligentie. De reden hiervoor is omdat de wereld van nature veel onzekerheid bevat. Denk aan de volgende punten:

- Kennis (we kunnen niet alles van de wereld weten)
- Incomplete modellen (verschijnselen die niet onder het model vallen)
- Sensoren (we kunnen de wereld enkel observeren met de tools die beschikbaar zijn en deze zijn meestal nog foutgevoelig.)
- Acties (we kunnen niet elke actie uitvoeren)

Als we spreken over werken met onzekerheid bedoelen we het opstellen van een hypothese, en deze hypothese (zo goed mogelijk) bewijzen aan de hand van het gegeven bewijs van de wereld. Er zijn 3 belangrijke kwesties in verband met het werken met onzekerheid:

- Representeren van onzekerheid
- Redeneren over onzekerheid
- Leren aan de hand van onzekerheid

Het representeren van onzekerheid gebeurd in het model. Een model is een weergave van een onzekerheidsprobleem waarbij elke wereld kan gesimuleerd worden. Bij het redeneren over onzekerheid maken we gebruik van het model om vragen te stellen over mogelijke hypotheses. (vb. Wat is de kans dat we een eerlijke muntstuk hebben als we deze 20 keer tossen en 15 keer hoofd en 5 keer munt verkrijgen.) In dit voorbeeld is de hypothese of het muntstuk eerlijk is. Het bewijs dat we hebben is dat we 20 keer tossen en 15 keer hoofd en 5 keer munt kregen. Wat we willen is de kans dat de hypothese klopt m.a.w. de kans dat het muntstuk eerlijk is.

Omdat het modelleren van een onzekerheidsprobleem, het redeneren over dit probleem en het leren aan de hand van de redeneringen een intensief proces is, is het efficienter om computerkracht hiervoor te gebruiken. Hierdoor werden er Probabilistische programmeertalen ontworpen.

## 1.1 Probabilistische programmeertalen

Probabilistische programmeertalen (of PPL's van Probabilistic Programming Languages) zijn programmeertalen met 2 hoofddoelen in zicht:

- Het vergemakkelijken van het modelleer proces
- Redeneren over het model

Er zijn PPL implementaties die een volledig nieuwe taal hebben geïmplementeerd speciaal voor het modelleren en redeneren, maar de meesten zijn geïmplementeerd als extensie van een bestaande programmeertaal.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Figuur 1.1: Bayes' rule geeft de kans dat een Hypothese A waar is gegeven het bewijs B.

### 1.1.1 Modelleren

### 1.1.2 Inferentie

Eén van de belangrijkste formules in het werken met onzekerheid is de Bayes' rule (figuur 1.1). Bayes' rule geeft de kans dat een hypothese waar is in een wereld waar er al dan niet bewijs is over deze wereld. PPl's kunnen hetzelfde berekenen aan de hand van een inferentie proces dat de taal implementeert. Elke PPL heeft een methode om de inferentie te bereken. Hoe ze dit doen verschilt voor elke PPL. Meer info over de inferentie methodes in de sectie 1.1.3 en 1.1.4.

Het berekenen van de inferentie is een zeer krachtig, maar zeer kostelijk proces qua rekenkracht. Veel PPL's zoeken een balans tussen hoe efficient ze inferentie kunnen berekenen en welke problemen ze kunnen modelleren (hoe expressief de taal is).

### 1.1.3 ProbLog2

ProbLog is een speciale vorm van PPL. Het is een Probabilistische logische programma PLP. ProbLog is gebasseerd op Prolog. Terwijl het niet echt een extensie is van Prolog heeft het wel dezelfde syntax. Het verschil met Prolog zit hem in de kansen dat voor de feiten of predicaten kunnen gezet worden. In Prolog is een feit altijd waar in het programma. In ProbLog is een feit waar met kans P en niet waar met kans (1-P). hetzelfde geldt voor predicaten. Als we een feit schrijven zonder kans is deze waar in het logische programma. Als we een feit schrijven als P::f dan is de feit f waar met kans P en niet waar met kans (P-1).

#### inferentie

De inferentie in ProbLog gebeurd aan de hand van verschillende processen:

1. converteren van logisch programma gewogen booleaanse formules.
2. inferentie proces op de gewogen booleaanse formules.

Deze stappen zijn onafhankelijke processen en zijn zo gekozen dat de output van het ene algoritme in het volgende algoritme past.

om te converteren van logisch programma gewogen booleaanse we 3 input's nodig. Het ProbLog model, het bewijs van het model en de vragen die men aan het model wilt stellen. Tijdens dit proces wordt het domein van de probabilistische variabelen opgesteld en deze variabelen zullen enkel de termen in hun domein gebruiken om het gewogen booleaanse formula op te bouwen.

De inferentie is anders afhankelijk van wat er gevraagt wordt.

- Als de MARG/SUCC gevraagt wordt maakt het systeem gebruik van een algoritme genaamd Weighted Model Counting (WMC).
- Als de MPE gevraagt wordt maakt het systeem gebruik van een algoritme genaamd MAX-SAT.

Dit zijn welbestudeerde algoritmes en voor meer informatie kunt u het artikel (Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas) lezen.

### 1.1.4 Anglican

**inferentie**

### 1.1.5 Stappenplan

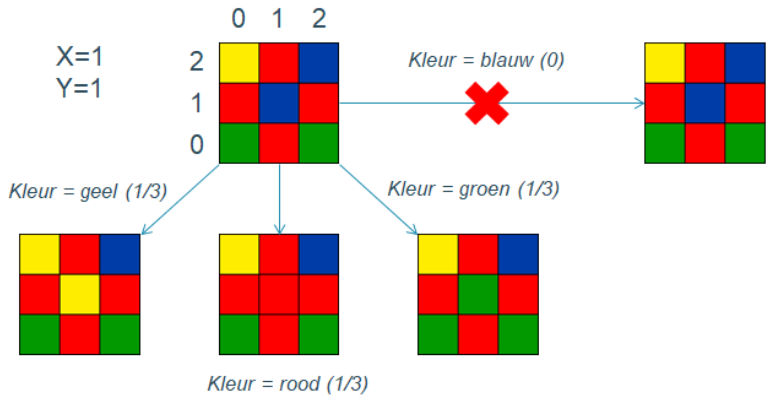
Om PPL's te evalueren en vergelijken maak ik gebruik van een case study. Ik gebruik een case study omdat het evalueren van programmeer talen die niet dezelfde programmeer paradigma hebben niet triviaal is. De case die ik gebruik is een zelf verzonnen spel dat probabilistische aspecten bevat. Dit spel modelleer ik in ProbLog en Anglican. Na de implementatie kan ik beginnen met het vergelijken en evalueren van de PPL's aan de hand van verschillende kwalitatieve en kwantitatieve criteria.

#### Spel

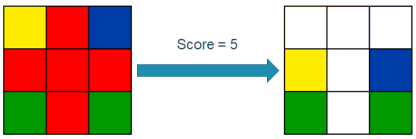
Het spel bestaat uit een bord van 10 op 10 blokken. Elke blok in het bord heeft 1 toegewezen kleur. Er zijn 4 kleuren die aan een blok kunnen toegewezen worden: rood, groen, geel, blauw.

Elke beurt kan de speler op 1 van de blokken drukken. Als er op een blok wordt gedrukt, verandert deze van kleur. De kleur waar de blok in verandert hangt af van welke kleur de blok had voor dat er op gedrukt werd.

- Een rode blok verandert met 1/3 kans in een groene blok, 1/3 kans in een blauwe blok en 1/3 kans in een gele blok.
- Een groene blok verandert met 1/3 kans in een rode blok, 1/3 kans in een blauwe blok en 1/3 kans in een gele blok.
- Een blauwe blok verandert met 1/3 kans in een rode blok, 1/3 kans in een groene blok en 1/3 kans in een gele blok.




Figuur 1.2: In de figuur gebruik ik een 3x3 bord, als er op de middelste blauwe blok (1,1) wordt gedrukt, verandert de blok met 1/3 kans in een rode blok, met 1/3 kans in een groene en met 1/3 kans in een gele blok. De blauwe blok zal nooit in een blauwe blok veranderen.




Figuur 1.3: Stel dat de blok in figuur 1.2 rood werd, dan staan er 3 blokken met dezelfde kleur naast elkaar. Het spel verwijdert deze blokken en voor elke blok die verwijdert is krijgt de speler een punt. In dit geval heeft de speler 5 punten. De gele en de blauwe blok vallen naar beneden tot ze op een andere blok of op de bodem vallen.

- Een gele blok verandert met 1/3 kans in een rode blok, 1/3 kans in een groene blok en 1/3 kans in een blauwe blok.

Figuur 1.2 geeft een visuele weergave van wat er gebeurd als er op een blok wordt gedrukt.

Als er drie of meer blokken van dezelfde kleur horizontaal/verticaal naast elkaar liggen verdwijnen ze en dit levert punten op. De blokken die zich boven de verdwenen blokken bevinden vallen naar beneden tot ze op een andere blok belanden ofwel op de bodem van het spelbord. De bedoeling van het spel is om in tien beurten zoveel mogelijk punten te behalen waarin de speler in elke beurt één blok van kleur kan veranderen. De beurt eindigt wanneer er geen 3 blokken van dezelfde kleur meer horizontaal en/of verticaal naast elkaar staan.

### Modelleren

### Evaluatiecriteria

De evaluatiecriteria kunnen gecategorizeerd worden in qualitatieve en quantitatieve criteria. Ook zijn er objectieve criteria en subjectieve criteria waar ik mijn evaluatie op basseer.

# Hoofdstuk 2

# Resultaten

# Hoofdstuk 3

# Conclusie

**Faculteit Computerwetenschappen**
Geel Huis, Kasteelpark Arenberg 11 bus 2100
3001 LEUVEN, BELGIË
tel. + 32 16 32 14 01
www.kuleuven.be