# 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:

1. Het modelleren van een wereld met onzekerheid
2. 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 PPL’s 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 PPL’s 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 PPL’s beschikbaar zijn de laatste jaren is het niet altijd duidelijk welke voordelen of nadelen ze hebben ten opzichte van andere PPL’s. Verschillende van deze PPL’s 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 PPL’s aan de hand van hun vorige iteratie of PPL’s met hetzelfde programmeerparadigma (Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas, LUC DE RAEDT et al). Deze artikels evalueren PPL’s met hetzelfde programmeerparadigma zoals logische probabilistische programmeertalen of functionele probabilistische programmeertalen.

In deze thesis ga ben ik van plan PPL’s met een verschillend programmeerparadigma zoals ProbLog en Anglican te evalueren. ik ga deze PPL’s 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 PPL’s aan de hand van vooropgestelde criteria. Ten slotte volgt een evaluatie van de PPL’s ten opzichte van elkaar. Uiteindelijk wil ik kunnen aantonen welke PPL het best presteert in welke criteria aan de hand van het opgegeven probleem.

# Achtergrond

Onzekerheid in artifici\"{e}le intelligentie is een van de invloedrijkste domeinen van artifici\"{e}le 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.

## Probabilistische programmeertalen

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

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

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

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.

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).

# Probleemstelling

## Probleem verzinnen

Als onzekerheidsprobleem heb ik gekozen om een spel te modelleren dat onderheven is aan probabilistische aspecten. Om het spel te spelen heb ik 4 spelstrategien ontwikkeld die elks ook onderheven zijn aan probabilistische aspecten.

### Spel

Het spel bestaat uit een bord van 10 op 10 blokken. Wanneer het spel gestart wordt krijgen de blokken een random kleur toegewezen maar er kunnen geen 3 van dezelfde blokken op een rij staan (enkel verticaal en horizontaal). Er zijn 4 kleuren in totaal: rood, groen, geel, blauw. In figuur 1 ziet u een voorbeeld van een begin bord.

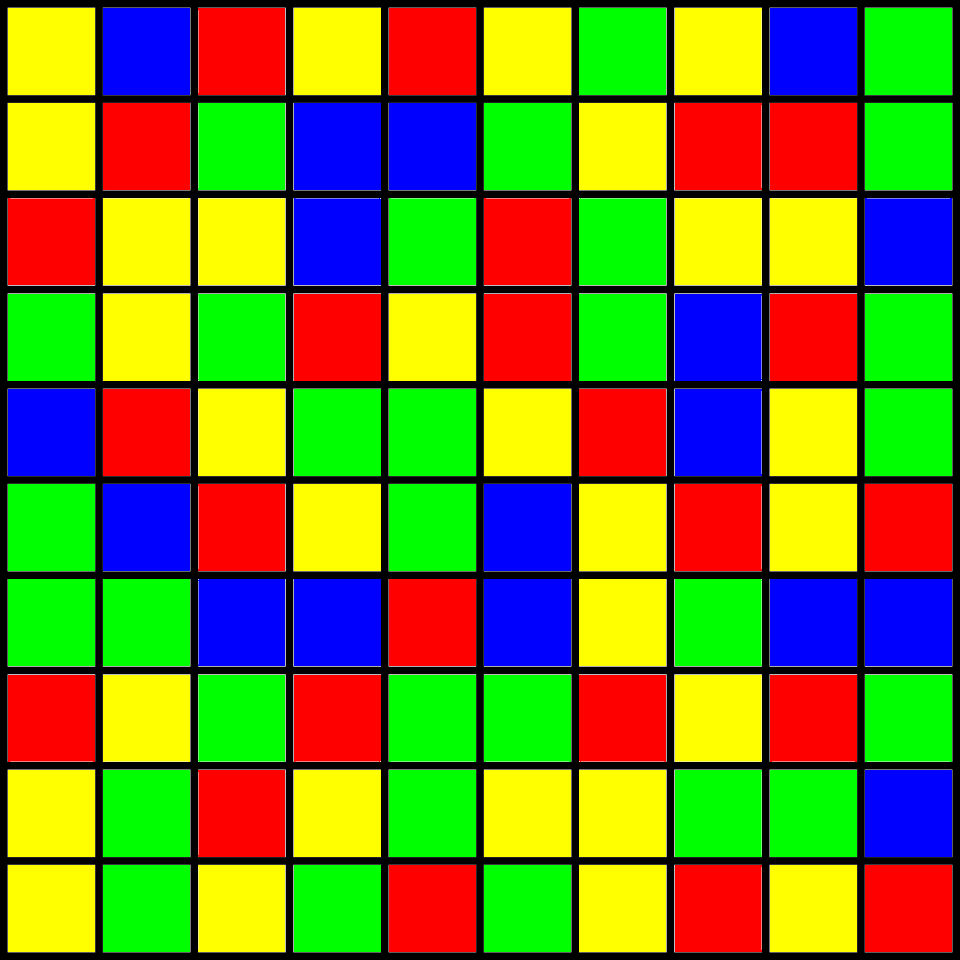


Figure voorbeeld van een intieel bord

De speler kan op elk van de blokken op het bord drukken. Als de speler op een blok drukt verandert deze van kleur. De kleur waar de blok in veranderd is random hangt af van een probabilistische distributie. Om het simpel te houden gebruik ik hier een uniforme distributie:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Verandert in  Block kleur | Rood | Groen | Blauw | Geel |
| Rood | 0 | 1/3 | 1/3 | 1/3 |
| Groen | 1/3 | 0 | 1/3 | 1/3 |
| Blauw | 1/3 | 1/3 | 0 | 1/3 |
| Geel | 1/3 | 1/3 | 1/3 | 0 |

Figure probabilistische distributie voor het veranderen van kleuren.

In woorden betekent dit dat als er op een rode blok wordt gedrukt er 1/3 kans is dat deze blok in een groene verandert, 1/3 kans in een blauwe verandert en 1/3 kans in een gele verandert.

Als er drie of meer blokken van dezelfde kleur ofwel horizontaal naast elkaar liggen ofwel 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. Voor elke blok die verwijdert wordt krijgt de speler 1 punt. De bedoeling van het spel is om in 5 beurten zoveel mogelijk punten te behalen waarin de speler in een elke beurt 1 blok van kleur mag veranderen. De beurt eindigt wanneer er geen 3 blokken van dezelfde kleur meer op een rij staan. In figuur 3 ziet u het verloop van een beurt in een 10x10 bord.

# 

Figure er wordt een blok gekozen om op te drukken, in dit geval een rode blok. De blok verandert met een 1/3 kans in een groene blok. Omdat er meer als 2 blokken van dezeflde kleur op een rij staan worden deze verwijdert en de bovenstaande blokken vallen naar beneden. Dit wordt herhaald tot er niet meer als 2 blokken van dezelfde kleur op een rij staan

### Strategien

Ik heb 4 strategien ontwikkeld om het spel te spelen.

* Uniforme strategy
* Kleuren ratio strategy
* Mogelijke score strategy
* Gewogen score strategy

Deze strategien kunnen gebruikt worden om het spel te spelen aan op een bepaalde wijze. Elke strategy kiest altijd 1 blok uit de mogelijke blokken die beschikbaar zijn. Welke blok dit is hangt van de strategy af.

#### Uniforme strategy

Als de uniforme strategy wordt toegepast is de kans dat een blok gekozen wordt uniform voor elke blok. Voor een 10x10 bord is de kans dat een blok wordt gekozen . In figuur 4 zien we alle mogelijke keuzes die de uniforme strategy kan kiezen in het gegeven bord.

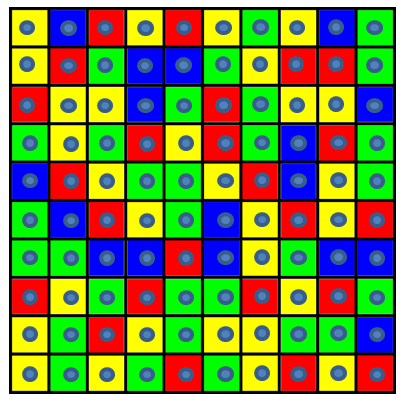


Figure in de uniforme strategy kunnen alle blokken gekozen worden met een uniforme kans verdeling. De kans is 1/100 voor elke blok in dit geval

#### Kleuren ratio strategy

Voor de kleuren ratio strategy worden eerst alle blokken met dezelfde kleur opgeteld. Uit de kleur met het minst aantal blokken wordt uniform een blok gekozen. In figuur 5 zien we dat de blauwe blokken in de minderheid zijn. De strategy zorgt ervoor dat er uniform een blauwe blok wordt gekozen.

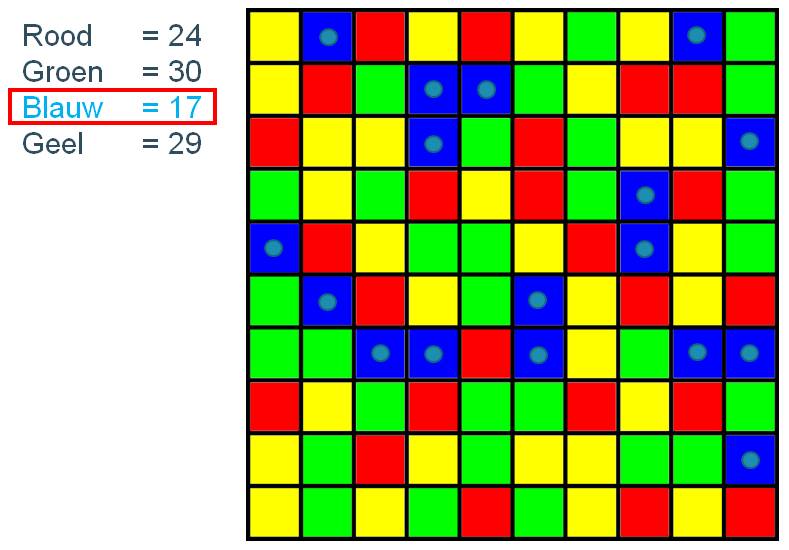


Figure In bovenstaande figuur is de kleuren ratio voor de blauwe blokken het minst. Hier wordt uniform een blauwe blok gekozen met een 1/17 kans voor elke blok

#### Mogelijke score strategy

In de mogelijke score strategy wordt elke voor elke blok appart na gegaan of deze een mogelijke score kan hebben. Een blok kan een mogelijke score hebben als deze blok kan veranderen in een kleur waarbij als de blok verandert in deze kleur, het een score oplevert. In figuur 6 ziet u alle blokken aangeduidt die een mogelijke score kunnen opleveren. Er wordt 1 blok uit deze blokken gekozen met een uniforme kansverdeling.

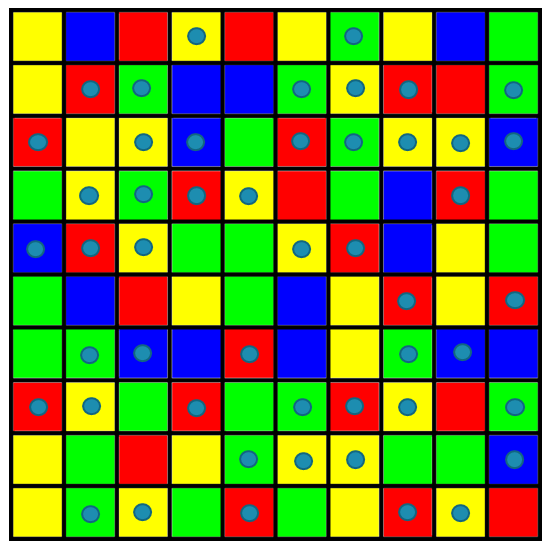


Figure In een mogelijke score strategy word er uniform een blok gekozen uit alle blokken met een mogelijke score. In dit geval zijn er 48 blokken met een mogelijke score dus de kansverdeling is 1/48 voor elke blok

#### Gewogen score strategy

De gewogen score strategy is een uitbreiding op de mogelijke score strategy waar we niet enkel naar de mogelijke score zien, maar naar de gewogen score. Elke blok kan veranderen van kleur aan de hand van een kansverdeling. De gewogen score strategy houdt rekening met deze kansverdeling. De gewogen score wordt berekent aan de hand van de score als een blok in Rood/Groen/Blauw/Geel verandert gewogen met de kans dat de blok verandert int Rood/Groen/Blauw/Geel.

# Gerelateerd werk

* Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas
* Probabilistic Logic Programming Concepts
* Exploiting Local and Repeated Structure in Dynamic Bayesian Networks
* Bayesian Reasoning and Machine Learning

# Conclusie

## Planning

1. Verdere evaluatie van ProbLog2
2. Implementatie van model in Anglican
3. Evaluatie van Anglican
4. Evaluatie van ProbLog2 en Anglican ten opzichte van elkaar
5. (optioneel) Stap 2 tot 4 herhalen voor een andere PPL.