Spel om beter te leren logisch redeneren

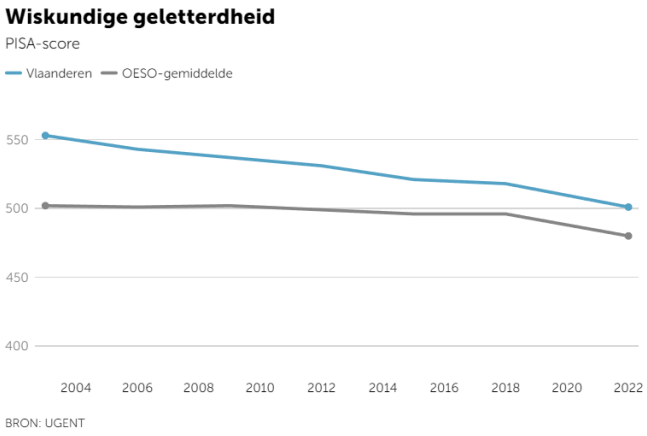
Wietze Schryvers , **Quinten Vandenschrick** ,
Katholieke Universiteit Leuven
wietze.schryvers@student.kuleuven.be, quinten.vandenschrick@student.kuleuven.be

Abstract

Vandaag de dag zijn er heel wat studies over de algoritmes voor het minimaliseren van logische uitdrukkingen. Deze paper bouwt hierop verder en analyseert hoe dergelijke problemen voor een gegeven moeilijkheid consistent gegenereerd worden. Deze methode kan gebruikt worden om de problemen te genereren voor het spel dat we ontwikkelden om beter te leren logisch redeneren. De manier waarop deze problemen consistent gegenereerd kunnen worden, werd empirisch onderzocht door te bepalen wat de moeilijkheidsgraad van de problemen onderscheidt. Dit onderzoek resulteerde in een methode om symmetrie toe te voegen of te verwijderen in de canonieke expressie, waardoor het aantal termen in de minimale uitdrukking binnen het gewenste interval kan worden gebracht.

1 Introductie

De laatste jaren kwamen in de media steeds meer berichten over de achteruitgang van het onderwijs. Verder toont het meest recente PISA-onderzoek aan dat er zich een dalende trend voordoet bij de wiskundige vaardigheden van jongeren [Droogenbroeck, 2023].



Figuur 1: grafiek van wiskundige vaardigheden van jongeren

Deze trend zou ook een afname van het logisch redeneren kunnen impliceren. Daarom hebben we een spel ontwikkeld dat logisch redeneren bevordert aan de hand van minimalisatie van logische functies. Er wordt de gebruiker aangeraden hiervoor Karnaugh-kaarten te gebruiken [Rushdi, 1986]. Deze methode wordt aangeraden omdat het een zeer interessante methode is waarbij patronen moeten gevonden worden om te minimaliseren. Dit zorgt voor extra inzicht in de logische uitdrukkingen, wat op zijn beurt dan weer bijdraagt aan het algemeen logisch redeneren. Om het logisch redeneren te kunnen bevorderen, is er nood aan verschillende niveaus van moeilijkheid in het spel. Deze paper bestudeert dan ook hoe verschillende moelijkheidsniveaus voor het minimaliseren van logische uitdrukkingen gegenereerd kunnen worden.

2 Achtergrond

2.1 Minimalisatie van logische uitdrukkingen

Zoals eerder vermeld staat het minimaliseren van logische functies centraal in het spel. Dit probleem wordt eerst kort geïllustreerd aan de hand van een voorbeeld. Stel dat er een vast aantal logische inputvariabelen (True of False) zijn, die gecombineerd worden tot een logische output (True = 1 of False = 0), dan kan dit voorgesteld worden in een tabel. Neem als voorbeeld volgende tabel (figuur 2), waarbij er 3 inputvariabelen zijn: x_1, x_2 en x_3. De combinatie van de drie inputvariabelen kan voorgesteld worden in een elementaire product term (p-term). Een p-term is een term in een logische uitdrukking die bestaat uit een product (AND-operatie) van variabelen. De p-term van een inputcombinatie wordt genoteerd als een product van alle variabelen, waarin negaties aangeduid worden met een ‘’‘.

	x_1	x_2	x_3	Output	p-termen
0	0	0	0	0	$x'_1x'_2x'_3$
1	0	0	1	1	$x'_1x'_2x_3$
2	0	1	0	1	$x'_1x_2x'_3$
3	0	1	1	1	$x'_1x_2x_3$
4	1	0	0	0	$x_1x'_2x'_3$
5	1	0	1	0	$x_1x'_2x_3$
6	1	1	0	1	$x_1x_2x'_3$
7	1	1	1	1	$x_1x_2x_3$

Figuur 2: waarheidstabel

De verzameling van alle combinaties van inputs die tot een output 1 leiden, kunnen voorgesteld worden in de volgende canonieke vorm:

$$True = x_1'x_2'x_3 + x_1'x_2x_3' + x_1'x_2x_3 + x_1x_2x_3' + x_1x_2x_3$$

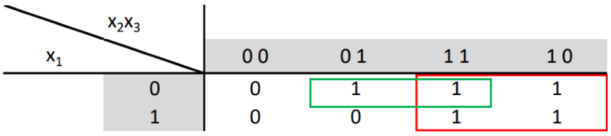
Uit de tabel kan eenvoudig afgeleid worden dat de output True is als x_2 = 1, of als x_1 = 0 en x_3 = 1. De uitdrukking kan dus vereenvoudigd worden tot:

$$True = x_2 + x_1'x_3$$

In de informatica wordt gebruik gemaakt van deze logische uitdrukkingen om elektrische schakelingen te beschrijven. De 'functie' van input variabelen naar output variabelen wordt in dit geval een circuit-transmissie genoemd. Het minimaliseren van deze uitdrukkingen voor schakelingen is belangrijk omdat dit tot besparingen leidt van de componenten die nodig zijn om de schakeling te kunnen realiseren. Hierdoor zullen de circuits ook efficiënter zijn.

2.2 Karnaugh kaarten

Een eenvoudig te begrijpen methode om logische uitdrukkingen te minimaliseren is met behulp van Karnaugh-kaarten. Deze methode werd gevonden in 1953 door Maurice Karnaugh. Om logische uitdrukkingen te vereenvoudigen, zijn er vaak moeilijke berekeningen vereist, maar met behulp van Karnaugh-kaarten hoeft dit niet. Deze methode toont op een visuele manier welke termen best gecombineerd worden. In wat volgt zal de werking van deze methode worden uitgediept aan de hand van de waarheidstabel uit figuur 2.

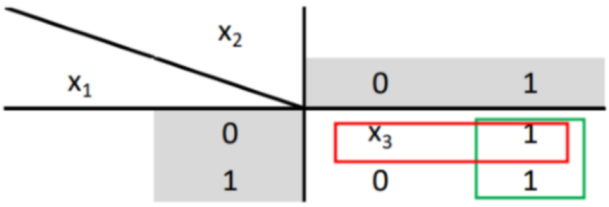


Figuur 3: voorbeeld: Karnaugh kaarten

Eerst worden de inputvariabelen in 2 groepen verdeeld. Vervolgens wordt er een tabel gemaakt met een van de groepen in de kolom wat hier de groep van x_2 en x_3 is en een van de groepen dat de rijen gaat vertegenwoordigen, wat hier dan x_1 is (figuur 3). De kaart wordt dan ingevuld met de waarheidswaarde van uit de tabel. Wanneer er twee of meer variabelen in een groep zitten, moeten deze gesorteerd worden volgens Gray code. Dit betekent dat twee opeenvolgende getallen altijd maar één bit verschillen. De volgende stap is om alle eentjes in de Karnaugh-kaart te groeperen in rechthoeken. Het aantal elementen in een rechthoek moet altijd een macht van 2 zijn waarbij de macht groter dan 0 moet zijn (2^n met $n > 0$). De rechthoeken moeten zo groot mogelijk gekozen worden om zo'n minimaal mogelijke uitkomst te bekomen. Meerdere rechthoeken mogen ook overlappen. Wanneer alle rechthoeken gevormd zijn, moet voor elke rechthoek de 'gemeenschappelijke inputs' gezocht worden. Dit zijn de inputcombinateis die in de hele rechthoek constant zijn. Vervolgens worden deze inputwaardes opgenomen in de p-term van de rechthoek. De som van al deze termen stelt de geminimaliseerde uitdrukking voor. Zo stelt het rode vierkant alle inputs voor waar x_2 = 1 en de groene rechthoek alle inputs waar x_1 = 0 en x_3 = 1. Dit resulteert opnieuw in de volgende minimale uitdrukking $x_2 + x_1'x_3$ zoals hierboven intuïtief werd gevonden.
Een nadeel van Karnaugh-kaarten is dat het vanaf meer dan 4 variabelen al snel complex wordt en het vaak efficiënter is om alternatieve algoritmes te gebruiken. Hiervoor zijn later varianten gevonden op de methode van Karnaugh-kaarten die beter werken met meer variabelen zoals Variable Entered Karnaugh Maps (VEKM).

Variable-entered Karnaugh maps

Een van de belangrijkste verbeteringen aan het algoritme van Karnaugh is het toelaten van variable-entered maps. Dit houdt in dat de variabelen niet allemaal apart worden voorgesteld in de tabel, maar ook in de cellen van andere geplaatst kunnen worden. De variable-entered map procedure van Fletcher kan toegepast worden op veel algemene transmissiefuncties. Deze methode laat zelfs toe dat de functie niet volledig gespecifieerd moet zijn, zowel in de kaart als in de entered variabelen. Nog een verdere uitbreiding werd voorgesteld door Rushdi [Rushdi, 1986]. Deze uitbreiding kan nog met meer variabelen werken omdat de map-entries ook meerdere variabelen mogen bevatten.



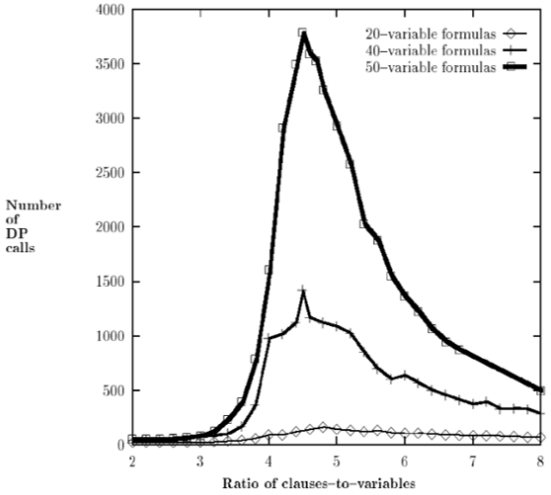
Figuur 4: voorbeeld: variable-entered Karnaugh map

Figuur 4 is een voorbeeld van een variable-entered Karnaugh-kaart waar opnieuw de waarheidstabel uit figuur 2 wordt gebruikt. De key-variabelen zijn in dit voorbeeld x_1 en x_2. Deze worden genoteerd in de rijen en kolommen. De variabele x_3 is hier gekozen als map-entered variabele. De cel linksboven betekent dat wanneer x_1 en x_2 nul zijn dat $x_3 = 1$ moet zijn om tot een output True te komen.

3 Eerder onderzoek

Vandaag de dag zijn er al heel wat studies rond de algoritmes voor het minimaliseren van logische uitdrukkingen uitgevoerd. Het onderzoek bestudeerde hoe de algoritmes in elkaar zitten en hun tijdscomplexiteit [Hoang-Gia Vu, 2021]. Echter zijn er geen studies die zich verdiepen in de moeilijkheidsgraad van een gegeven logische uitdrukking. Daarentegen werd er al wel onderzoek uitgevoerd naar de moeilijkheid van SAT-problemen. Het SAT-probleem is een beslissingsprobleem, met als invoer een logische propositie. Het probleem bestaat uit de vraag of er een logische

toekenning (True of False) voor de variabelen bestaat zodanig dat de volledige propositie waar is. Dit wordt kort geïllustreerd met een voorbeeld. Neem de propositie $(p_1 \vee \neg p_3) \wedge \neg p_2 \wedge (p_3 \vee p_4)$. Wanneer p_1 en p_4 de toekenning *True* en p_2 de toekenning *False* krijgt is de volledige propistie ook *True*. Deze propositie is dus vervulbaar (*satisfiable*).



Figuur 5: grafiek van moeilijkheid van SAT-probleem

Uit het onderzoek over de moeilijkheid van SAT-problemen [David Mitchell, 1992] bleek dat voor functies die kort of lang zijn, er redelijk snel een oplossing gevonden wordt, maar voor de formules van middelmatige lengte duurt het veel langer om ze op te lossen. Dit verschijnsel wordt ook wel het easy-hard-easy patroon genoemd. De verklaring [David Mitchell, 1992] voor dit patroon is dat uitdrukkingen met weinig beperkingen onderbeperkt zijn en veel mogelijke toewijzingen hebben. Zo wordt een toewijzing snel gevonden. Formules met zeer veel beperkingen zijn overbeperkt (en meestal niet oplosbaar), daardoor worden contradicties gemakkelijk ontdekt waardoor er snel gezegd kan worden dat dit onoplosbaar is (*unsatisfiable*).De uitdrukkingen die daartussen zitten zijn veel moeilijker omdat ze weinig (of geen) toekenningen hebben. Het minimaliseren van logische uitdrukkingen vertoont veel gekijkenissen met het SAT-probleem. Bijgevolg werd er een gelijkaardig resultaat verwacht in dit onderzoek.

4 Probleemstelling

Voor het realiseren van een spel dat gebruik maakt van minimaliseren van logische uitdrukkingen, is het belagrijk dat de gebruiker verschillende moeilijkheidsniveaus aangeboden kan krijgen. Er moeten dus willekeurige logische uitdrukkingen gegenereerd kunnen worden van verschillende niveaus. Verder is het belangrijk dat de canonieke expressie, die de gebruiker krijgt, te minimaliseren valt. De canonieke expressie moet dus meer p-termen hebben dan de minimale uitdrukking. Stel dat het aantal p-termen in de canonieke vorm genoteerd wordt met p, en het aantal p-termen in de minimale uitdrukking genoteerd wordt met r, dan moet gelden dat

$$r \leq c_1 * p$$

Om het spel interessant te houden is het ook belangrijk dat de gebruiker niet alle p-termen uit de canonieke expressie kan verwerken in aanzienelijk minder p-termen. Hiervoor nemen we dat voor p en r zoals hiervoor beschreven werd, ook moet gelden dat

$$r \geq c_2 * p$$

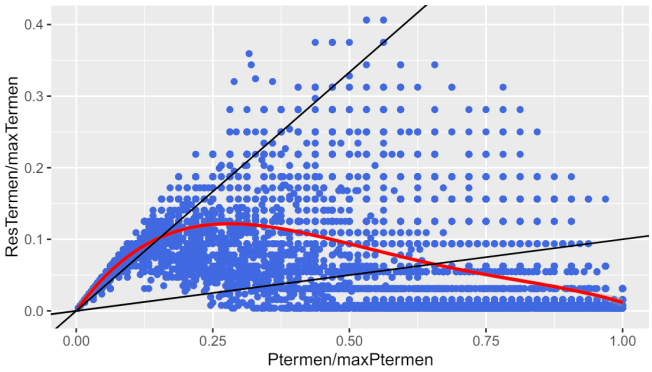
In dit onderzoek wordt gekozen voor $c_1 = \frac{2}{3}$ en $c_2 = \frac{1}{10}$. Hiervoor is er een methode nodig om deze problemen consistent te kunnen genereren. Bijgevolg is de onderzoeksvraag : "Hoe kan men de problemen van minimalisatie van logische uitdrukkingen met een gewenst aantal termen in de minimale uitdrukking genereren?".

5 Methodologie

Het onderzoek werd aangepakt op een empirische wijze. Concreet werd dit empirisch onderzoek uitgevoerd door willekeurige logische uitdrukkingen te genereren die variëren in het aantal p-termen en het aantal variabelen. De logische uitdrukkingen worden gegenereerd in de canonieke vorm. Dit wil zeggen dat elke term in de uitdrukking voor alle variabelen een toekenning heeft (True of False). Elke uitdrukking met n variabelen heeft dan maximaal 2^n termen in de canonieke expressie. De generator neemt eerst een willekeurig aantal variabelen. Vervolgens wordt een willekeurig getal k gekozen tussen 1 en 2^n, dit zijn het aantal termen die de expressie zal bevatten. Hierna worden alle mogelijke termen gegenereerd met het gekozen aantal variabelen. Uit deze lijst met termen worden willekeurig k termen gekozen. Daarna wordt het algoritme van Quine-McCluskey gebruikt om deze uitdrukking te minimaliseren. Hierover wordt allerlei data verzameld zoals: het aantal p-termen in de canonieke expressie, aantal ptermen in de minimale expressie, aantal variabelen, gemiddeld aantal variabelen per eindterm en de uitvoeringstijd. Deze resultaten werden bestudeerd om te bepalen wat de factoren zijn waardoor een bepaalde uitdrukking moeilijker is dan een andere uitdrukking. In het onderzoek werd gebruik gemaakt van het feit dat uitdrukkingen, die met veel p-termen na minimalisatie overblijven, moeilijker zijn om te minimaliseren. Als er namelijk veel p-termen zijn na minimalisatie dan betekent dat, dat er veel verbanden zijn gezocht bij het toepassen van Karnaugh-kaarten, wat zorgt voor meer werk om te minimaliseren. De tijd die het algoritme nodig had, volstaat niet als moeilijkheidsgraad. Het algoritme duurt langer wanneer er meer p-termen zijn in het probleem. Uitdrukkingen waar elke inputcombinatie True geeft, en dus een maximaal aantal p-termen heeft (voor dat aantal variabelen), zouden lang duren, terwijl deze heel makkelijk te minimaliseren zijn (True = True). Het aantal p-termen in de minimale expressie wordt daarom gezien als moeilijkheidsgraad. In deze analyse werden daarom verbanden gezocht tussen de onderzoeksvariabelen en het aantal p-termen in de minimale expressie.

6 Resultaten

Figuur 6 is het resultaat van het onderzoek. De grafiek toont de verhouding tussen het aantal p-termen in de canonieke vorm p, ten opzichte van het aantal p-termen in de minimale uitdrukking r. Beide worden gedeeld door het maximaal aantal p-termen in de canonieke vorm. Voor n variabelen zijn dit er 2^n. In de grafiek valt een merkwaardige vorm te zien. De grafief stijgt eerst tot een gegeven punt en gaat erna terug dalen (figuur 6). Deze grafiekvorm vertoont gelijkenissen met het eerder besproken easy-hard-easy patroon dat in het onderzoek over SAT werd waargenomen.



Figuur 6: grafiek resultaten

De verklaring hiervoor is dat wanneer de canonieke vorm weinig p-termen bevat, de uitdrukking weinig tot niet te minimaliseren valt. Bijgevolg zijn deze problemen niet interessant om tijdens het spel aan een gebruiker voor te leggen zoals besproken in de probleemstelling. Wanneer de canonieke vorm veel p-termen bevat (en dus veel van de inputcombinaties *True* geven), kan deze uitdrukking vaak zeer sterk geminimaliseerd worden wat resulteert in minder minder termen in de minimale uitdrukkingen. De meerderheid van de interessante problemen, die voldoen aan de eisen zoals gedefiniëerd in de probleemstelling, voldoen aan:

$$0,15 \leq \frac{p}{2^n} \leq 0,65$$

In de grafiek is te zien dat er in dit interval nog steeds problemen zijn die te weinig of te veel geminimaliseerd kunnen worden. Hiervoor is er een methode nodig die dit probleem oplost.

7 Methode voor het genereren

Het spel zorgt voor verschillende moeilijkheidsgraden door meer variabelen toe te voegen. Het maximaal aantal p-termen in de canonieke expressie stijgt exponentieel met het aantal variabelen. Zo zijn er voor n variabelen maximaal 2^n p-termen. Dit zorgt er ook voor dat het maximaal aantal p-termen in de geminimaliseerde uitdrukking exponentieel stijgt. Dit geldt voornamelijk wanneer het aantal p-termen in de canonieke vorm p voor n variabelen voldoet aan

$$p/2^n \leq 0,7$$

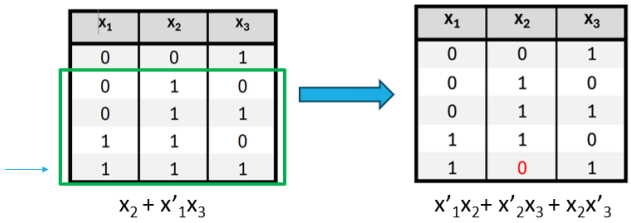
Voor een probleem te genereren wordt er eerst een willekeurig probleem gegenereerd met het aantal p-termen in de canonieke vorm in het interval $[0,15 * 2^n; 0,65 * 2^n]$. Vervolgens wordt dit probleem geminimaliseerd en wordt er gekeken naar het aantal p-termen in het resultaat. Wanneer voor het aantal p-termen in het resultaat minder p-termen r zijn geldt dat $r \leq 0,1 * p$, dan bevat het probleem te veel symmetrie. Met symmetrie wordt bedoeld de mate waarin een zelfde toekenning van een variabele (True of False), of combinaties van toekenningen van meer variabelen, veel voorkomt. Veel symmetrie zorgt in dit geval dat veel termen uit de canonieke vorm samengenomen kunnen worden in één term in de geminimaliseerde uitdrukking, waardoor deze weinig p-termen bevat. Omgekeerd wanneer er geldt dat $r \geq 0,66 * p$, bevat het probleem te weinig symmetrie. Om het tekort of het teveel aan symmetrie in het probleem op te lossen wordt de oplossing van het huidige probleem gebruikt om meer symmetrie toe te voegen of te verwijderen uit het probleem. Hiervoor wordt er een term gekozen uit het probleem waarin één of meer variabelen tegengestelde toekenningen krijgen, zodat dit zorgt voor meer of minder symmetrie.

7.1 Verwijderen van symmetrie

Voor het verwijderen van symmetrie wordt in de oplossing gezocht naar de meest minimale termen. Dit zijn de termen in de minimale uitdrukking die zo min mogelijk toekenningen hebben voor variabelen, en bijgevolg meeste aantal p-termen uit de canonieke vorm bevatten. Neem bijvoorbeeld $x_2 + x_1' x_3$ (figuur 7) hierin is x_2 de meest minimale term (Algorithm 1).

Algorithm 1 vind kandidaten(oplossing)

```
minToekenningen ← ∞
resultaat ← [ ]
for term in oplossing do
    toekenningen ← aantal toekenningen in term
    if toekenningen < minToekenningen then
        minToekenningen ← toekenningen
        resultaat ← [term]
    else if toekenningen = minToekenningen then
        voeg term toe aan resultaat
return resultaat
```



Figuur 7: Voorbeeld: verwijderen van symmetrie

Vervolgens wordt er een subset bepaald van p-termen uit de canonieke vorm, die voldoen aan de meest minimale term. In dit voorbeeld zijn dat de vier termen $x_1x_2x_3$, $x_1x_2x_3'$, $x_1'x_2x_3$, $x_1'x_2x_3'$. Hieruit wordt nu willekeurig een term gekozen, waarin één van de variabelen die nog een toekenning heeft in de meest minimale term, wordt geflipt (Algorithm 2). In dit geval kan dus enkel variabele x_2 geflipt worden in de gekozen term. Dit zorgt ervoor dat de term x_2 in het aangepaste probleem niet meer deel kan uitmaken van de minimale uitdrukking.

Algorithm 2 Verwijderen van symmetrie(kandidaten)

```
for kandidaat in kandidaten do
    canTermen ← termen bevat door kandidaat
    for term in canTermen do
        variabelen ← variabelen in kandidaat
        for variabele in variabelen do
            nieuweTerm ← flip variabele in term
            if deze expressie bevat nieuweTerm niet then
                verwijder term uit expressie
                voeg nieuweTerm toe aan expressie
                return
    ▷ Geen flip gevonden die nog niet in de expressie zit
Kies random kandidaat
canTermen ← termen bevat door kandidaat
verwijder random term uit canTermen
Voeg een andere random p-term toe
```

Convergentie

Voor de convergentie is het belangrijk om op te merken dat de verzameling termen die geflipt mag worden altijd minstens één term bevat. Er wordt steeds minstens één term gevonden in de minimale uitdrukking met het minst aantal toekenningen van variabelen (wanneer alle termen in de oplossing evenveel toekenningen hebben, dient het algoritme dus alle termen te bekijken). Bijgevolg zal er steeds een term gevonden kunnen worden die geflipt kan worden.
Het verwijderen van symmetrie zorgt ervoor dat een term in de oude oplossing vervangen wordt door minstens twee andere termen. Het aantal nieuwe termen hangt af van het aantal p-termen uit de canonieke uitdrukking die voldoen aan de meest minimale term. Wanneer er n variabelen zijn en de meest minimale term bevat m variabelen met toekenning, dan bevat de meest minimale term 2^{n-m} p-termen uit de canonieke expressie. Wanneer één p-term hier wordt uitgehaald (of veranderd), dan wordt deze term opgesplitst in $n - m$ termen in de nieuwe minimale uitdrukking. De term bekomen door een variabele te flippen kan mogelijks wel symmetrie op een andere plek toevoegen.

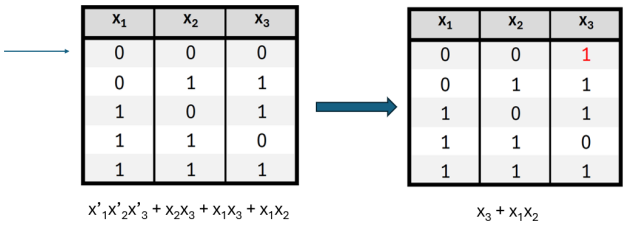
7.2 Toevoegen van symmetrie

Voor het toevoegen van symmetrie wordt in de oplossing gezocht naar de minst minimale termen. Dit zijn dus de termen in de minimale uitdrukking waarin het meest variabelen een toekenning hebben en bijgevolg het minst p-termen uit de canonieke vorm bevatten. Neem hiervoor bijvoorbeeld $x_1'x_2'x_3' + x_2x_3 + x_1x_3 + x_1x_2$ (figuur 8) hierin is $x_1'x_2'x_3'$ de minst minimale term. (Algorithm 3)

Algorithm 3 vind termen(oplossing)

```
maxToekenningen ← −1
resultaat ← [ ]
for term in oplossing do
    toekenningen ← aantal toekenningen in term
    if toekenningen > maxToekenningen then
        maxToekenningen ← toekenningen
        resultaat ← [term]
    else if toekenningen = maxToekenningen then
        voeg term toe aan resultaat
return resultaat
```

Vervolgens wordt er opnieuw een subset bepaald van p-termen uit de canonieke vorm, die voldoen aan de minst minimale term. Hieruit wordt een willekeurige term gekozen, t_1. In dit voorbeeld is dit opnieuw enkel de term $x_1'x_2'x_3'$. Deze wordt nu vergeleken met alle andere termen om een term t_2 te vinden. Deze moet op exact twee variabelen van t_1 verschillen. Uit deze twee variabelen waarin t_1 en t_2 van elkaar verschillen, wordt er willekeurig één van geflipt in de term t_1, wat term t_1' geeft. Belangrijk hierbij is na te kijken dat t_1' nog geen deel uit maakt van de expressie. In dit voorbeeld wordt de term $x_1'x_2'x_3'$ vergeleken met $x_1'x_2x_3$. Deze verschillen in variabelen x_2 en x_3. Hiervoor wordt x_3 gekozen om in de term $x_1'x_2'x_3'$ te flippen. Dit resulteert in de term $x_1'x_2'x_3$.



Figuur 8: voorbeeld: toevoegen van symmetrie

Algorithm 4 Toevoegen van symmetrie(kandidaten)

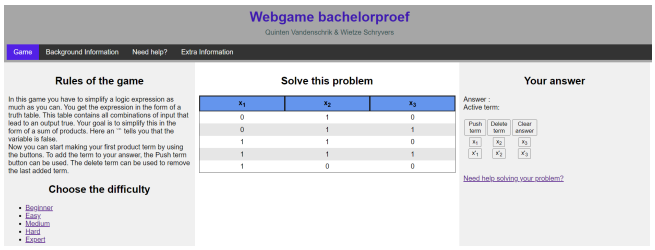
```
for kandidaat in kandidaten do
    for term2 in canonieke expressie do
        verschilIndexen ← kandidaat verschilt van term2
        if lengte verschilIndexen = 2 then
            for index in verschilIndexen do
                nieuweTerm ← flip x_index in term
                if deze expressie bevat nieuweTerm niet then
                    verwijder term uit expressie
                    voeg nieuweTerm toe aan expressie
                    return
    ▷ Geen flip gevonden die nog niet in de expressie zit
Verwijder random kandidaat
Voeg een andere random p-term toe
```

Convergentie

Voor de convergentie is het belangrijk op te merken dat de verzameling termen die geflipt mag worden altijd minstens één term bevat. Dit volgt uit de manier waarop deze termen worden bepaald. Er wordt steeds minstens één term uit (een kopie van) de oplossing gehaald. Bijgevolg omvat deze oplossing minstens één originele p-term minder. Als dit niet het geval is, zou de term die uit de oplossing gehaald is, niet essentiëel zijn. Dit is in contradictie met dat de oplossing minimaal was. Er is dus altijd minstens één term die aangepast kan worden. Deze term kan zo gekozen worden dat er zo min mogelijk andere symmetrie verwijdert wordt. Het algoritme zal dus altijd een stap kunnen nemen.

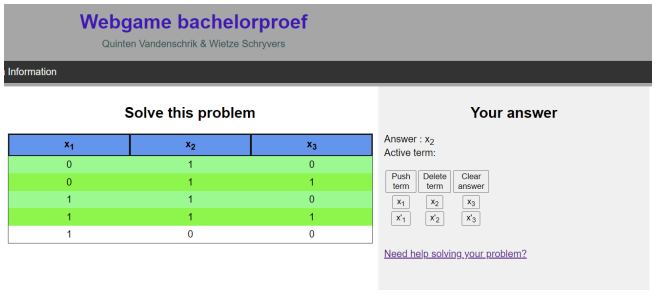
8 Het spel

Via onderstaande link kan het spel geopend worden en de werking ervan uitgetest worden. Wanneer de gebruiker de site opent, zal die op het startscherm uitkomen (figuur 9). Van hieruit kan het gewenste niveau uit de verschillende moeilijkheidsniveaus gekozen worden. Ook kan er achtergrondinformatie geraadpleegd worden op de website. https://wietzeschr.github.io/webgame-bachelorproef/html/game.html



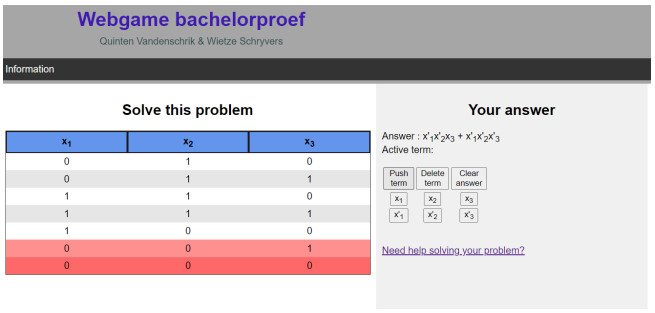
Figuur 9: webgame

Het spel werkt interactief mee met de gebruiker. Wanneer de gebruiker een term toevoegt aan zijn oplossing, dan kleurt het spel al de termen van de orginele uitdrukking groen die de toegevoegde term bevat.



Figuur 10: webgame: groen kleuren wanneer term bevat

Wanneer de gebruiker een term toevoegt aan zijn oplossing die enkele input combinaties bevat, die niet deel uit maken van de canonieke expressie, dan voegt het spel de overbodige termen toe aan de tabel en kleurt deze rood.

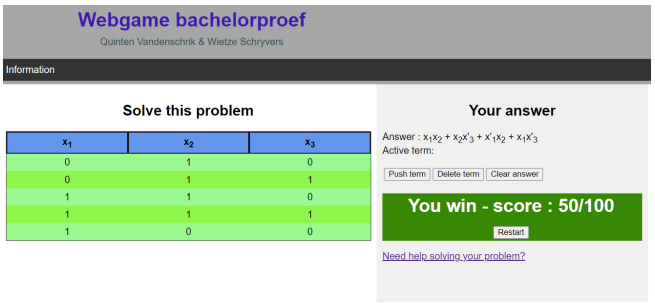


Figuur 11: webgame: overbodige termen toegevoegd

Wanneer de oplossing van de gebruiker alle input combinaties bedekt van het probleem en geen input combinatie te veel bevat wordt er weergegeven dat het probleem opgelost is. De gegeven oplossing wordt beoordeeld aan de hand van een score op 100. Deze score wordt berekend aan de hand van het aantal AND en OR poorten dat nodig zou zijn om deze uitdrukking te implementeren in een elektrische schakeling. Het aantal poorten in de canonieke vorm wordt aangeduid met p, het aantal poorten in de minimale uitdrukking wordt aangeduid met m en het aantal poorten in het antwoord wordt aangeduid met a. De score wordt dan berekent als volgt:

$$\frac{p-a}{p-m} * 100$$

Er kan enkel een score van 100/100 gehaald worden als de oplossing van de gebruiker even minimaal is als de meest minimale oplossing.



Figuur 12: webgame: scoresyteem

9 Conclusie

Deze paper had als doel een methode te vinden voor het consistent genereren van logische uitdrukkingen met een gegeven moeilijkheidsgraad. Aan de hand van empirisch onderzoek heeft deze studie aangetoond dat wanneer de verhouding van het aantal p-termen bij start en het maximaal aantal p-termen tussen de 0,15 en 0,65, ligt de kans het hoogst is uitdrukkingen te genereren met de gewenste minimalisatiegraad. Door vervolgens symmetrie toe te voegen of te verwijderen in de canonieke expressie kan het gewenste aantal p-termen in de minimale uitdrukking gerealiseerd worden voor de gewenste moeilijkheidsgraad. Met deze methode voor het generen van logische

uitdrukkingen, hebben we een goedwerkend spel ontwikkeld dat helpt beter leren logisch redeneren. Onderzoekers zouden in de toekomst deze studie kunnen repliceren, maar dan voor probablistische problemen. De probablistische problemen kunnen gebruik maken van een bayesiaans netwerk met veel gelijke kansen in de voorwaardelijke probablistische tabel. Dan zou deze tabel gecomprimeerd kunnen worden met de methodes die bestudeerd zijn in dit onderzoek en kan er een gelijkaardig spel ontwikkelt kunnen worden. Meer specifiek zouden ze kunnen onderzoeken of er een gelijk aardig patroon kan gevonden worden voor probablistische problemen.

Dankwoord

We zouden bij deze graag professor Luc De Raedt willen bedanken voor het geven van het interessante onderwerp. Ook bedanken we Vincent Derkinderen en Jaron Maene voor het goed begeleiden tijdens het onderzoek.

Referenties

[David Mitchell, 1992] Hector Levesque David Mitchell, Bart Selman. Hard and Easy Distributions of SAT Problems. https://www.researchgate.net/publication/2769087_Hard_and_Easy_Distributions_of_SAT_Problems, 1992.

[Droogenbroeck, 2023] Kelly Van Droogenbroeck. Zware achteruitgang van Vlaamse onderwijs PISA-onderzoek demorgen.be. https://www.demorgen.be/snelnieuws/opnieuw-zware-klap-voor-het-vlaamse-onderwijs-leerlingen-boeren-st 2023.

[Hoang-Gia Vu, 2021] Anh-Tu Nguyen ThanhBangLe Hoang-Gia Vu, Ngoc-Dai Bui. Performance Evaluation of Quine-McCluskey Method on Multi-core CPU. http://eprints.lqdtu.edu.vn/id/eprint/10375/1/main_camera_ready.pdf, 2021.

[Rushdi, 1986] Ali M. Rushdi. Improved variable-entered Karnaugh map procedures. https://www.sciencedirect.com/science/article/abs/pii/0045790687900218?via%3Dihub, 1986.