Algoritme en complexiteit SAT solver

Micha de Groot

October 2016

1 Introductie

In dit korte verslag wordt de implementatie van een SAT solver in Python besproken. Om de berekende oplossing te verifiëren wordt een ander Python programma gebruikt dat voor deze opdracht gemaakt is. De input voor de solver is van de volgende vorm, eerst een regeld ie het aantal variabelen en clause, gevolgd door alle variabelen en clauses:

```
8 18
R1
G1
B1
Y1
R2
B2
G2
Y2
R1 B1 G1 Y1
~R1 ~B1
~R1 ~G1
~R1 ~Y1
~B1 ~G1
~B1 ~Y1
~G1 ~Y1
R2 B2 G2 Y2
~R2 ~B2
~R2 ~G2
~R2 ~Y2
~B2 ~G2
~B2 ~Y2
~G2 ~Y2
~R1 ~R2
~B1 ~B2
~G1 ~G2
~Y1 ~Y2
```

2 Eerste poging: Resolutie

In de eerste poging werd er gepoogd dmv resolutie (`https://nl.wikipedia.org/wiki/Resolutie_(logica)`) de formule te vereenvoudigen waardoor het aantal variabelen waarvoor een waarheidswaarde moet worden berekent zal verkleinen. Nadat er verscheidene problemen met de implementatie opgelost waren bleek echter dat resolutie in dit geval niet een juiste aanpak is. Alleen voor bepaalde logische formules is resolutie toepasbaar.

3 Tweede poging: Recusieve aanpak

In de tweede poging is er qua algoritme weer van de grond af begonnen. De eenvoudige aanpak is bij mogelijkheid $\{x_0 = 0, x_1 = 0, ..., x_n = 0\}$ beginnen en kijken of het klopt. Zo niet, wordt de binaire waarde van de set met één verhoogd. De waarde dan weer evalueren. Herhalen tot er een correcte waarheidswaarde gevonden is.
Deze aanpak is erg inefficiënt en is niet geïmplementeerd. Om tijd te besparen is er gelijk nagedacht over een betere aanpak. Men observeert dat een toewijzing van waarheidswaarden verworpen kan worden voordat alle clausules of variabelen geëvalueerd zijn. Als uit variabele 1 tot i blijkt dat de waardes inconsistent zijn, kunnen alle mogelijke waardes voor $i + 1$ tot n voor dezelfde waardes van 1 tot i verworpen worden.
Met deze gedachte in het achterhoofd is het volgende recursieve algoritme bedacht. Het idee er achter is, is dat er niet gelijk een hele set aan waarheidswaardes wordt gemaakt maar stapsgewijs een waarde wordt toegewezen aan variabele $i + 1$ als blijkt dat de set waarheidswaardes van 1 tot i consistent is met de formule.

formule is een logische formule in CNF. variabelen is een set met de variabelen uit formule met waarde true, false of none.

```
assignRecursive(formule, variabelen)
    if: formule inconsistent met variabelen
        return False
    if: geen none in variabelen
        return True
    tempVariabelen <- variabelen (met op eerste none een 1)
    try: assignRecursive(formule, tempVariabelen)
    if True:
        variabelen[eerste none] = 1
        return True
    else:
        tempvariabelen[wat eerst none was] = 0
        try: assignRecursive(formule, variabelen)
        if True:
```

```
            variabelen[eerst none] = 0
            return True
        else:
            return False
```

Met dit algoritme worden in de meeste gevallen minder opties geprobeerd dan met de naïve probeer-alles functie. Namelijk, als het recursieve algoritme dieper in de zoekboom gaat, is de kans dat er weer ver terug moet worden gebacktracked klein. Bij de naïve functie kunnen alle opties van de eerste i variabelen geprobeerd worden terwijl de $i + 1$'ste variabele een waardde heeft waarvoor de formule sowieso niet vervulbaar is.

Met deze implementatie kost het ongeveer 0.07 seconden om voor een formule met 30 variabelen en 270 clauses een juiste aanwijzing van waarheidswaardes te vinden, en ongeveer 0.04 seconden voor een formule met acht variabelen en 18 clauses.