

Opgave Constraint Processing

De deadline voor het indienen van je verslag is **woensdag 9 december, 12u**. We verwachten je verslag op papier in de studentenbrievenbus in 200A.

In dit project zullen we een aantal constraint problemen voorstellen en ze oplossen met een constraint solver die hoort bij *SICStus Prolog*. Deze solver gebruikt een hybride algoritme, dat *backtracking* combineert met *arc consistency checking*. Concreet wordt een constraint probleem hierdoor dus—conceptueel gezien—als volgt opgelost. Initieel wordt een arc consistency algoritme, zoals bijvoorbeeld AC3, uitgevoerd. Als dit niet voldoende is om een oplossing te vinden, wordt er aan één van de variabelen een waarde toegekend uit zijn nog overgebleven domein, waarna het AC3 algoritme opnieuw wordt uitgevoerd. Als ook dit nog geen oplossing oplevert, wordt er aan een volgende variabele een waarde toegekend, waarna opnieuw AC3 wordt uitgevoerd, enzoverder. Indien er op een bepaald moment een inconsistentie optreedt (het domein van een variabele wordt leeg), dan wordt er over deze toekenningen *gebacktrackt*, net zoals dit in de cursus voor *forward checking* en *lookahead checking* beschreven staat.

In tegenstelling tot de cursus legt deze solver geen beperkingen op aan het aantal variabelen dat in een constraint mag voorkomen. Ook constraints tussen méér dan twee variabelen zijn met andere woorden toegelaten. De algoritmes die de solver gebruikt zijn dan ook niet precies de algoritmes uit de cursus, maar wel uitbreidingen hiervan die ook dergelijke constraints aankunnen.

We behandelen drie problemen met deze solver.

Op volgende URL vind je de software die je voor deze opgave moet gebruiken:

<http://www.cs.kuleuven.be/~maartenm/constraints/>

Instructies voor het gebruik hiervan vind je op de webpagina zelf.

1 Magisch vierkant: 2x2

Een magisch vierkant van orde n is een n -op- n vierkant waarin de getallen van 1 tot en met n^2 zodanig gerangschikt staan dat de som van alle kolommen, rijen en de twee diagonalen dezelfde is. Dit is bijvoorbeeld een magisch vierkant van orde 5, waarin deze som telkens 65 is:

11	24	7	20	3
4	12	25	8	16
17	5	13	21	9
10	18	1	14	22
23	6	19	2	15

Elke 1-op-1 vierkant is triviaal magisch. Bestaan er magische vierkanten van orde 2? Beantwoord deze vraag met behulp van de tool. Gebruik hiervoor variabelen A , B , C en D , die als volgt de inhoud van de vakjes van het vierkant voorstellen:

A	B
C	D

In je verslag dien je voor deze opgave, en alle volgende, steeds te **tonen welke constraints je hebt ingevoerd** zoals deze verschijnen in het veldje rechtsonderaan.

2 Magisch vierkant: 3x3

Als tweede opgave gaan we op zoek naar een magisch vierkant van orde 3. Gebruik hiervoor een geïndexeerde variable V , die als volgt de inhoud van de vakjes van het vierkant voorstelt:

$V(0)$	$V(1)$	$V(2)$
$V(3)$	$V(4)$	$V(5)$
$V(6)$	$V(7)$	$V(8)$

Geef opnieuw de constraints die je hebt ingevoerd.

3 Sudoku

Sudoku is een populair puzzelspel, waarbij het erop aankomt een rooster op correcte wijze in te vullen met cijfers van 1 tot en met 9. Concreet moet elk cijfer precies één keer voorkomen in elke rij, elke kolom, en in elk van de negen kleine drie-op-drie vierkantjes. In bijlage vind je een aantal dergelijke puzzels, onderverdeeld in 2 moeilijkheidsniveaus (gemakkelijk—moeilijk). Doe hiermee het volgende:

- Stel een sudoku voor als constraint-probleem en encodeer je voorstelling in de constraint solving tool. Los hiermee de puzzels in bijlage op. Merk je een verschil bij het oplossen van problemen van verschillende moeilijkheidsgraad op te lossen? **Let wel:** De tool rapporteert telkens zijn *volledige* run-tijd; dit is de tijd die het gekost heeft om zowel alle indices te overlopen om de gepaste constraints te creëren, als om deze constraints op te lossen. Een betere manier om te beoordelen hoe moeilijk het constraint

probleem is, is daarom om te kijken naar het aantal keer dat er tijdens het oplossen van het constraint probleem een conflict is opgetreden (dwz. er was een variabele met een leeg domein), waardoor er *backtracking* nodig was. Ook dit wordt door de tool gerapporteerd. Kan je je observaties verklaren vanuit je kennis over constraint solving?

- De tool biedt verschillende heuristieken aan. Vergelijk de **leftmost** met de **first-fail + constraint count** heuristiek. Welke presteert het beste? Is er ook hier een verschil tussen de verschillende moeilijkheidsniveaus? Verklaar zoveel mogelijk je observaties.
- Bij het oplossen van een sudoku zal een menselijke speler vaak bepaalde strategieën toepassen. Op volgende website:

<http://www.sudokudragon.com/sudokustrategy.htm>

vinden we bijvoorbeeld als 3^e strategie de “*two out of three rule*.” Voeg aan je formalizatie een constraint toe die deze strategie expliciet encodeert. Heeft dit een invloed op de gevonden oplossingen? Beïnvloedt dit de tijd die nodig is om een sudoku op te lossen? Kunnen we ook hier weer een invloed waarnemen van de gebruikte heuristiek of de moeilijkheidsgraag van de sudoku? Verklaar zoveel mogelijk je observaties.

HINTS:

De software die je ter beschikking hebt laat slechts één niveau van indexering toe. Dit betekent dat je niet kan spreken over, bijvoorbeeld, vakje (4, 5), maar dat je de vakjes in een puzzel elk één coördinaat moet toekennen, zoals bijvoorbeeld in volgende tekening getoond wordt:

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	.	.	.					
				.	.	.	79	80

Als je deze indexering gebruikt, kan je het rijnummer van een bepaald vakje terugvinden als:

$$Rij(i) = i / 9 \text{ (waarbij / de gehele deling is);}$$

het kolomnummer vind je dan weer terug als volgt:

$$Kolom(i) = i \text{ mod } 9 \text{ (rest bij gehele deling van } i \text{ door 9).}$$

Eens je een algemeen model van sudoku hebt, kan je natuurlijk een concrete opgave ingeven door het gelijkstellen van bepaalde variabelen aan bepaalde

waardes. Omdat dit echter snel nogal omslachtig wordt, bieden we je een andere webpagina aan, die dit vergemakkelijkt:

<http://www.cs.kuleuven.be/~maartenm/constraints/sudoku.php>

De instructies voor het gebruik hiervan zijn te vinden op de webpagina zelf. **Dit werkt enkel maar indien je de indexering van je variabelen doet zoals hierboven aangeven!**

Bij technische problemen met de software kan je mailen naar:
maarten.marien@cs.kuleuven.be.

Veel succes!

Gemakkelijke sudoku's:

	6	1						7
7	2	5				6	9	
		8	6	4			5	2
			2	1				9
9		2				3		1
1				9	6			
5	4			7	8	2		
	1	9				7	8	3
2						9	4	

						6	7	
4		6	9	7		3		
		5			3	9	1	8
9	4		6	5	8	7		
		3	2	1	7		4	9
5	9	4	7			8		
		8		3	4	2		5
	1	2						

	4	2		8			9	5
		3				8		2
8					2	1		
	5	7	6		4	9	3	
			3		7			
	6	4	2		8	5	1	
		6	9					1
9		8				7		
5	7			3		4	2	

Moeilijke sudoku's:

				8			9	
			6		4			1
	7	8					5	
8				4		5		
3			9		2			8
		4		1				7
	1					6	2	
5			1		9			
	4			6				

		6	5			8	3	
	7			8				
				6		4		1
7					4	3		
		8				7		
		2	6					5
8		5		4				
				2			9	
	6	7			9	5		

7			9		6			
		4			7		6	9
	1		2	4				
	5					7		
6								1
		2					8	
				2	4		5	
4	3		8			9		
			1		3			7