

## Gequoteerde zitting Prolog: Mainarizumu

NAAM:

RICHTING:

### Enkele praktische afspraken

- Je krijgt twee uur om deze opdracht **individueel** op te lossen.
- Je raadpleegt **enkel afgedrukte kopies** van de slides (eventueel met handgeschreven nota's) en de ingebouwde manual van SWI-Prolog (gebruik bv `?-help(write).` of `?-apropos(select).`)
- In de map **1819\_Gequoteerde/Prolog\_Woensdag** op Toledo vind je de bestanden `mainarizumufacts.pl`, `mainarizumufacts2.pl` en `run.pl`, evenals de indienmodule.
  - De bestanden `mainarizumufacts.pl` en `mainarizumufacts2.pl` bevatten de feiten voor een aantal voorbeelden. Het bestand `run.pl` kan uitgevoerd worden met `swipl -f run.pl` om je oplossing te toetsen aan de voorbeelden.
  - Als de opdracht expliciet de naam (en ariteit) van een predicaat vermeldt, ben je verplicht om dezelfde naam (en ariteit) te gebruiken in je oplossing.
  - Je oplossing zet je in een bestand `myprolog.pl` en de eerste lijnen van dit bestand moeten je naam, studentenummer en richting bevatten.

```
% Jan Jansen
% r0123456
% master cw
```
  - Na twee uur, of wanneer je klaar ben, dien je het `myprolog.pl` bestand in via Toledo.



In Figuur 1 zie je een voorbeeld van een spelletje ‘mainarizumu’. Dit spelletje wordt gespeeld in een vierkant rooster.

De bedoeling van het spel is om het  $n$  bij  $n$  rooster zodanig in te vullen dat:

- Elk getal tussen 1 en  $n$  exact één keer voorkomt per rij en per kolom.
- De ongelijkheden aangegeven tussen verschillende vakjes gerespecteerd worden; e.g. als in het vakje links boven een 2 staat mag in het vakje daar onmiddellijk rechts van enkel een 1 staan.
- Bij vakjes die een omcirkeld getal  $x$  delen, moet de inhoud van die twee vakjes exact  $x$  schelen.

Uiteraard heeft een spelletje mainarizumu een unieke oplossing. Figuur 2 toont de oplossing voor het spelletje mainarizumu van Figuur 1.

4		>		>	
		√			
3					
	√		√		
2		>		③	
1				>	
0			<		
	0	1	2	3	4

Figuur 1: Een mogelijke startsituatie van het spelletje ‘mainarizumu’.

We stellen een spelletje ‘mainarizumu’ voor met behulp van volgende Prolog feiten:

- `size(N)` stelt de grootte van het vierkante rooster voor.
- `gt((X1,Y1),(X2,Y2))` stelt voor dat er een  $>$  teken staat tussen het vakje op positie  $(X1,Y1)$  en het vakje op positie  $(X2,Y2)$ . Let op: het vakje linksonder wordt voorgesteld door positie  $(0,0)$ .

4	3	>	2	4	>	1	5
			√				
3	5		1	2		4	3
	√			√			
2	4	>	3	1		5	③ 2
1	1		5	3	>	2	4
0	2		4	<	5	3	1
	0	1	2	3	4		

Figuur 2: De unieke oplossing van het spelletje ‘mainarizumu’ getoond in Figuur 1.

- $\text{differ}((X1,Y1),(X2,Y2), V)$  stelt voor dat voor dat het vakje  $(X1,Y1)$  en het vakje  $(X2,Y2)$  het omcirkelde getal  $V$  delen. Bijv.  $\text{differ}((3,2),(4,2),3)$  voor Fig. 1.

**Opgelet:** Alle voorbeeldqueries in dit document werken met het rooster uit Figuur 1.

## 1 Kettingen

Wanneer we van een reeks van **2 of meer** vakjes weten dat elk vakje in de reeks kleiner is dan het vorige omdat ze  $>$ -tekens delen, dan spreken we van een ketting. Bij kettingen van lengte  $n$  ligt de waarde van elk vakje volledig vast.

Mogelijke voorbeelden van kettingen uit Figuur 1 zijn  $(1,4)$ ,  $(1,3)$  en  $(0,4)$ ,  $(1,4)$ ,  $(1,3)$ . Zoals je kan zien is de eerste voorbeeldketting een **deel van** de tweede voorbeeldketting. Kettingen die zelf **geen** deel uitmaken van een grotere ketting noemen we **maximale** kettingen.

### 1.1 Maximale kettingen vinden

Schrijf een predicaat `maximalChains/1` dat alle maximale kettingen in een ‘mainarizumu’ puzzel gedefinieerd door `size/1` en `gt/2` feiten teruggeeft (houd **géén** rekening met `differ/3` feiten). Stel een ketting voor als een

`chain(L)` term waarbij `L` een lijst van posities van vakjes is, waarbij de vakjes van groot naar klein gesorteerd staan. De maximale ketting  $(0,4)$ ,  $(1,4)$ ,  $(1,3)$  wordt bijvoorbeeld voorgesteld door `chain([(0,4), (1,4), (1,3)])`.

De volgorde van de `chain/1` termen in de lijst speelt geen rol.

```
?- maximalChains(Ch).
```

```
Ch = [chain([(2, 4), (3, 4)]), chain([(2, 3), (2, 2)]),
chain([(2, 1), (3, 1)]), chain([(2, 0), (1, 0)]),
chain([(0, 4), (1, 4), (1, 3)]), chain([(0, 3), (0, 2), (1, 2)])];
false.
```

Let op! Het is mogelijk dat maximale chains gedeeltelijk overlappen, volgende situatie is bijvoorbeeld mogelijk in een 6x6 rooster:

5						
4						
3			>	<		
2			<	>		
1						
0						
	0	1	2	3	4	5

Hierbij zijn 4 maximale kettingen:

- `chain([(1,3), (2,3), (2,2), (1,2)])`
- `chain([(1,3), (2,3), (2,2), (3,2)])`
- `chain([(3,3), (2,3), (2,2), (1,2)])`
- `chain([(3,3), (2,3), (2,2), (3,2)])`

## 2 Mainarizumu

Schrijf een predicaat `mainarizumu(Sol)` zodanig dat `Sol` een oplossing is van de mainarizumu puzzel gedefinieerd door de `size/1`, `gt/2`, en `differ/3` feiten.

Stel een oplossing voor als een lijst van `at/3` termen, waarbij `at(X,Y,V)` voorstelt dat op positie `(X,Y)` de waarde `V` moet staan. De volgorde van de `at/3` termen speelt geen rol.

```
?- mainarizumu(Sol).
```

```
Sol = [at(4, 4, 5), at(4, 3, 3), at(4, 2, 2), at(4, 1, 4), at(4, 0, 1),  
at(3, 4, 1), at(3, 3, 4), at(3, 2, 5), at(3, 1, 2), at(3, 0, 3),  
at(2, 4, 4), at(2, 3, 2), at(2, 2, 1), at(2, 1, 3), at(2, 0, 5),  
at(1, 4, 2), at(1, 3, 1), at(1, 2, 3), at(1, 1, 5), at(1, 0, 4),  
at(0, 4, 3), at(0, 3, 5), at(0, 2, 4), at(0, 1, 1), at(0, 0, 2)];  
false.
```

## 3 Eilanden

Een aaneengesloten gebied van vakjes  $v_1, \dots, v_m$ , noemen we een **eiland** wanneer het mogelijk is om tussen elk willekeurig paar van vakjes  $v_i$  en  $v_j$  een pad te vinden zodat:

- alleen vakjes van het eiland gebruikt worden, en
- voor elke overgang van een vakje  $v_a$  naar een vakje  $v_b$  geldt dat de waarde van  $v_a$  en  $v_b$  ten hoogste 1 verschillen.

We stellen een eiland voor als een Prolog term `island(Pos)` waarbij `Pos` een **geordende** lijst is van posities. Bijv. `island([(0,0),(0,1)])`.

### 3.1 Eilanden vinden

Schrijf een predicaat `islands(Board, (X,Y), Isle)` dat slaagt wanneer `Isle` het **grootste** eiland voorstelt dat de gegeven positie `(X,Y)` bevat op het speelbord voorgesteld door `Board`. Een speelbord wordt voorgesteld door

een lijst van at/3 feiten, zoals in sectie 2 wordt uitgelegd. Figuur 3 toont de eilanden op het speelbord van Figuur 2.

```
?- mainarizumu(Sol), island(Sol, (1,0), Isle).
```

```
Sol = ... %see above,  
Isle = island([(1,0),(1,1),(2,0)]);  
false.
```

4	3	2	4	1	5
3	5	1	2	4	3
2	4	3	1	5	2
1	1	5	3	2	4
0	2	4	5	3	1
	0	1	2	3	4

Figuur 3: De eilanden op het ‘mainarizumu’ veld getoond in Figuur 2.