Gequoteerde zitting Prolog: Kapotte Rekenmachine

NAAM: RICHTING:

Enkele praktische afspraken

- Je krijgt twee uur om deze opdracht **individueel** op te lossen.
- Je raadpleegt voor deze opgave **enkel**:
  - afgedrukte kopies van de slides (evt. met handgeschreven nota's)
  - de ingebouwde manual van SWI-Prolog (bvb. `?- apropos(select).`)
  - de opgaven en oplossingen van de oefenzittingen op Toledo.
- In de map **1516_Gequoteerde/Prolog_Donderdag** op Toledo vind je deze opgave en de indienmodule.
  - 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 `prolog.pl`. Bovenaan in dit bestand zet je je naam, studentennummer en richting.

    ```
    % Jan Jansen
    % r0123456
    % master cw
    ```

  - Na twee uur, of wanneer je klaar bent, dien je het `prolog.pl` bestand in via Toledo.

Inleiding

Stijn zit in de les wiskunde. Hij wil een getal op zijn rekenmachine intypen, maar ontdekt dat zijn rekenmachine kapot is. Er werken slechts een paar knoppen meer. Enkele cijfers, de + en *-knop werken nog. Hij vraagt aan jou om uit te vissen hoe hij het getal dat hij wil, te zien krijgt op zijn rekenmachine. De rekenmachine start met een 0 op het scherm. We zoeken de sequentie van getallen en operatoren die tot het gevraagde getal leiden.

1 Een eerste oplossing

Aangezien enkel de + en * knoppen werken, is het nooit nuttig om een operatie te doen met een getal dat groter is dan het gezochte getal. Het volgende predicaat representeert alle getallen die we kunnen intypen met de gegeven cijfers, onder een gegeven maximum.

Opdracht 1 Schrijf een predicaat `relevantNumber/3` zodat het predicaat waar is als het laatste argument een getal is dat je kunt maken met cijfers uit de lijst in het eerste argument en dit getal kleiner of gelijk is aan het tweede argument. De volgorde van de oplossingen is niet relevant. Zorg er wel voor dat er geen dubbele oplossingen. Denk eraan dat 0 een mogelijk cijfer kan zijn.

```
?- relevantNumber([0,1],30,X).
X = 0 ;
X = 1 ;
X = 10 ;
X = 11.

?- relevantNumber([3,4],44,X).
X = 3 ;
X = 4 ;
X = 33 ;
X = 43 ;
X = 34 ;
X = 44.
```

Opdracht 2 Nu zijn we klaar om een oplossing te schrijven voor Stijns probleem. Schrijf een predicaat `calcul/4` dat, gegeven de cijfers die nog werken en het huidige cijfer op het display, de operaties vindt die Stijn nog moet uitvoeren om aan het doelgetal te komen. Het eerste argument is de lijst van bruikbare cijfers, het tweede argument is het doelgetal, het derde argument is het getal dat nu op het display staat, en het laatste argument is de lijst met operaties die Stijn moet doen. In die lijst staan getallen (eventueel van meerdere cijfers) en de operaties + en *.

Hint: Toets een operatie in, toets een **relevant getal** in, kijk wat dat al als resultaat heeft. Zoek vervolgens naar een reeks operaties die de oplossing vindt vanaf dit nieuwe startpunt. Maak gebruik van staartrecursie.

```
?- calcul([3,7],10,1,X).
X = [*, 3, +, 7] ;
X = [*, 7, +, 3] ;
X = [+, 3, +, 3, +, 3] ;
false.

?- calcul([2],12,6,X).
X = [*, 2] ;
X = [+, 2, +, 2, +, 2] ;
false.

?- calcul([1,2],33,11,X), length(X,L), L < 5.
X = [*, 2, +, 11],
L = 4 ;
X = [+, 1, +, 21],
L = 4 ;
X = [+, 11, +, 11],
L = 4 ;
X = [+, 21, +, 1],
L = 4 ;
X = [+, 22],
L = 2 ;
false.
```

Hoogstwaarschijnlijk leidt je naieve oplossing tot een stack overflow als Stijn cijfer 0 of 1 gebruikt. Pas je predicaat `calcul/4` aan zodat:

- Je nooit met 0 vermenigvuldigt.
- Je nooit 0 met iets anders vermenigvuldigt.
- Je nooit met 1 vermenigvuldigt.
- Je nooit ergens 0 bij optelt.

```
?- calcul([1,0],10,1,X).
X = [*, 10] ;
X = [+, 1, +, 1, +, 1, +, 1, +, 1, +, 1, +, 1, +, 1, +, 1] ;
false.
```

2 De beste oplossing

Stijn is blij dat je hem geholpen hebt maar hij is nog steeds niet tevreden. Hij zou graag de antwoorden gesorteerd hebben volgens hoeveel moeite het kost om ze in te typen. Je mag veronderstellen dat elk cijfer en elke operator hem even veel moeite kosten. We passen eerst de oplossing van Opdracht 2 aan zodat we kunnen zoeken naar een oplossing met een gegeven kost.

Opdracht 3 Schrijf een predicaat `calculCost/5` dat, op de voorlaatste plaats een extra argument heeft: het exacte aantal toetsen dat Stijn wil intypen om de oplossing te vinden.

```
?- calculCost([3],33,1,3,X).
X = [*, 33] ;
false.

?- calculCost([1,4],20,0,6,X).
X = [+, 1, +, 4, *, 4] ;
X = [+, 4, *, 4, +, 4] ;
X = [+, 4, +, 1, *, 4] ;
false.
```

Opdracht 4 Schrijf een predicaat `sortedCalcul/3` dat, gegeven de cijfers die Stijn kan gebruiken en het doelgetal, de oplossingen voor Stijn zijn probleem vindt, gesorteerd volgens hun kost. De rekenmachine start met het getal 0 op het display. Binnen dezelfde kost mag de volgorde arbitrair zijn. Doe dit door `calculCost/5` aan te roepen met een steeds groter wordend getal. De maximale kost om een getal te bereiken is twee keer dat getal.

```
?- sortedCalcul([2,3],35,X).
X = [+, 2, +, 33] ;
X = [+, 3, +, 32] ;
X = [+, 32, +, 3] ;
X = [+, 33, +, 2] ;
X = [+, 2, *, 2, *, 3, +, 23] ;
...

?- sortedCalcul([7],210,X).
X = [+, 7, *, 7, +, 7, +, 77, +, 77] ;
X = [+, 7, *, 7, +, 77, +, 7, +, 77] ;
...
```

3 Betrapt!

Stijn is blij met je werk en slaagt erin om de wiskundeleerkracht af te leiden met zijn getokkel. Helaas geeft je programma niet altijd een antwoord en wordt Stijn betrapt door de leerkracht. Die leerkracht geeft hem het volgende raadsel: welke getallen tussen 1 en 30 kun je allemaal niet bereiken op je rekenmachine?

Opdracht 5 Schrijf een predicaat `unreachable/2` dat, gegeven de cijfers die Stijn kan gebruiken, de getallen tussen 1 en 30 teruggeeft die Stijn niet op zijn rekenmachine kan tonen.

```
?- unreachable([5,6],Y),write(Y).
[1,2,3,4,7,8,9,13,14,19]
Y = [1, 2, 3, 4, 7, 8, 9, 13, 14|...].

?- unreachable([2,3],Y),write(Y).
[1]
Y = [1].
```