Gequoteerde zitting Prolog: Compressie

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 **1617_Gequoteerde/Prolog_woensdag** op Toledo vind je de bestanden `compressionfacts.pl` en `run.pl`. Ook de indienmodule staat daar.
  - Het bestand `compressionfacts.pl` bevat de facts die gebruikt werden voor de voorbeeld queries van deze opdracht. Het bestand `run.pl` kan worden gebruikt m.b.v. `swipl -f run.pl` om de voorbeeldqueries uit te voeren. Dit bestand bevat ook de verwachte uitvoer voor deze queries.
  - 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` en de eerste lijnen van dit bestand moeten je naam, studentennummer en richting bevatten.

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

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

Door middel van compressie worden grote bestanden verkleind opgeslagen. Een manier om aan compressie te doen is het gebruik van een look-up table met veel voorkomende datasequenties.

Voor deze opgave krijg je een lookup table aangeleverd door middel van `code(Sequence, Code)` feiten (zoals bv. in Listing 1), waarbij `Sequence` een lijst prologtermen is en `Code` één enkele prologterm die gebruikt kan worden om de lijst voor te stellen in een compacte weergave. We gebruiken hier de conventie dat de prologtermen die voorkomen in `Sequence` altijd letters zijn, en de prologtermen die voorkomen als `Code` altijd getallen.

Listing 1: Een voorbeeld van een look-up table.

```
code([a,b], 1).          code([b,c], 4).
code([b,e], 2).          code([c,d],5).
code([b,c,d,e], 3).      code([a,b,c],6).
```

Opdracht 1 Schrijf een predicaat `decompress(Compressed, Decompressed)` dat gegeven een gecomprimeerde voorstelling `Compressed`, slaagt met de originele voorstelling als waarde voor `Decompressed`. Merk op dat een `Compressed` lijst ook termen kan bevatten die geen code zijn. In dit geval is dit een stuk van het originele bestand dat niet gecomprimeerd kon worden.

```
?- decompress([2], D).
D = [b,e].

?- decompress([1,3], [a, b, c, d, e]).
false.

?- decompress([a, 2, c, 3], D).
D = [a,b,e,c,b,c,d,e].
```

Opdracht 2 Nu willen we ook compressie voor onze rekening nemen door middel van een predicaat `compress(Uncompressed, Compressed)`. Dit predicaat slaagt als `Compressed` een **geldige** compressie is voor de lijst van

prologtermen voorgesteld door `Uncompressed` (Je mag ervan uitgaan dat deze altijd gegeven is).

Let op! Een compressie is **geldig** wanneer het mogelijk is om vanuit de compressie eenduidig de originele voorstelling te recupereren, en als de compressie alleen stukken uit de originele voorstelling bevat wanneer deze **noodzakelijk** zijn, i.e. wanneer vanaf deze positie in het originele bestand geen codewoord gevonden kan worden.

```
?- compress([a,b,c], [1,c]).
true. % [1,c] is een geldige, maar niet de meest
% compacte voorstelling van [a,b,c].
% [6] is ook een geldige en compactere voorstelling.

?- compress([a,b,c], [a,4]).
false.

?- compress([b,c,d,e], C).
C = [4, d, e];
C = [3].
```

Opdracht 3 Zoals uit opdracht 2 blijkt, is het mogelijk dat een enkel bestand meerdere compressies heeft. Sommige van deze compressies zijn minder efficiënt dan andere, omdat ze langer zijn. Schrijf een predicaat `optcompression(Uncompressed, Compressed)` dat slaagt wanneer `Compressed` (een van de) kortste gecomprimeerde voorstellingen is van `Uncompressed`.

```
?- optcompression([a,b,c,d,f], C).
C = [6, d, f];
C = [1, 5, f].

?- optcompression([b,c,d,e], C).
C = [3].

?- optcompression([a,b,c,d,e], [a, 3]).
false.
```