Künstliche Intelligenz - Übung 10

Julian Dobmann, Kai Kruschel

Aufgabe 1 Parser/DCG

Man kann den gegebenen Parser recht leicht um die Unterstützung von Multiplikation erweitern, indem man eine Regel (3) für das Zeichen `*` nach der Additionsregel hinzufügt.

Um die Klammerung von ganzen Expressions (und nicht nur von Termen) zu unterstützen, fügt man Regel (1) hinzu. Diese Regel wird als erste von `expr/1` eingesetzt, da Klammerung die allerhöchste Bindungsstärke besitzt.
Um die Bindungsstärke der Operatoren `+` und `*` korrekt zu berücksichtigen, wird Regel (3) eingefügt.

```
...
(1) expr(X)     --> "(", expr(X), ")".
(2) expr(add(mul(Y,Z), A)) --> term(Y), "*", term(Z), "+", expr(A).
(3) expr(mul(Y, Z)) --> term(Y), "*", expr(Z).
(4) expr(add(Y,Z)) --> term(Y), "+", expr(Z).
...
```

Aufgabe 2

Algebra 1

`algebra1/3` wird verwendet, um die Unsicherheitsschwelle von 20 umzusetzen. Das heißt für alle Regeln in der Datenbank, deren CF < 20 ist, wird der CF als 0 interpretiert. Das wird folgendermaßen erreicht:

```
algebra1(RuleCF, CF, AdjustedCF) :-
  CF >= 20,
  AdjustedCF is RuleCF.

algebra1(RuleCF, CF, AdjustedCF) :-
  fail.
```

Für den Fall, dass CF kleiner ist als 20 failt algebra1 einfach und prov als Konsequenz davon auch.

Algebra 2

`algebra2/3` wird verwendet, um ?

Algebra 3

`algebra3/3` wird verwendet, um aus zwei bestehenden Regeln mit gleichem Goal und unterschiedlichen CFs eine einzelne Regel abzuleiten.

```
% update the known CF
algebra3(CF, OldCF, NewCF) :-
  CF > 0, OldCF > 0,
  X is (CF + OldCF*(100 - CF)/100),
  int_round(X, NewCF).

algebra3(CF, OldCF, NewCF) :-
  CF < 0, OldCF < 0,
  X is -(-CF - OldCF *(100 + CF)/100),
  int_round(X, NewCF).

% does not divide by 0, because one of the Values has to be < 100
algebra3(CF, OldCF, NewCF) :-
  (CF < 0; OldCF < 0); (CF > 0; OldCF > 0),
  abs_minimum(CF, OldCF, MinCF),
  X is (100 * (CF + OldCF) / (100 - MinCF)),
  int_round(X, NewCF).
```