Introduction to Data and Knowledge Engineering SS2010 – 8. Tutorium

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgabe 8.6: Backgammon

a)

TECHNISCHE UNIVERSITÄT DARMSTADT

a) Schreiben Sie ein Prädikat dice (P, X, Y), das die Augenzahlen von dem Wurf des Spielers P modelliert.

```
dice(P,X,Y) :-  player(P,C), X = 1 , Y =1.
dice(P,X,Y) :-  player(P,C), X = 1 , Y =2.
...
dice(P,X,Y) :-  player(P,C), X = 6 , Y =1.
...
dice(P,X,Y) :-  player(P,C), X = 6 , Y = 6.
```

Alternative Lösung:

```
dice(X,Y):-  player(P,C), member(X,[1,2,3,4,5,6]),
             member(Y,[1,2,3,4,5,6]).
```

Aufgabe 8.6: Backgammon

Listen in Prolog

TECHNISCHE UNIVERSITÄT DARMSTADT

- Für Listen gibt es in Prolog eine eigene Schreibweise:
  - Elemente zwischen eckigen Klammern eingeschlossen.
  - durch Kommata getrennt.
  - Elemente sind beliebige Terme.

- member/2, bestimmt, ob ein Term in einer Liste enthalten ist.

Aufgabe 8.6: Backgammon

b)

TECHNISCHE UNIVERSITÄT DARMSTADT

b) Schreiben Sie ein Prädikat, das die möglichen Züge für Stein S zurückgibt.

```
color_dice(C,X,Y) :-  player_stone(P,S), dice(P,X,Y),
                      color(S,C).
free(C,POS) :-        player(P,C), can_move(P,POS).

moves_red(S,Z) :-     color_dice(red,X,Y), point(S,P),
                      Z is P + X, free(red,Z), Z < 25.
moves_red(S,Z) :-     color_dice(red,X,Y), point(S,P),
                      Z is P + Y, free(red,Z), Z < 25.
moves_red(S,Z) :-     color_dice(red,X,Y), point(S,P),
                      Z is P + X + Y, free(red,Z), Z < 25.
moves_red(S,Z) :-     color_dice(red,X,Y), point(S,P),
                      Z is P + X + X + X, free(red,Z), Z < 25.
```

Aufgabe 8.6: Backgammon

b)

TECHNISCHE UNIVERSITÄT DARMSTADT

Analog zu `moves_red(S,Z)` können wir die möglichen Züge für weiße Steine auch definieren.

```
moves(S,P) :-  moves_red(S,P).
moves(S,P) :-  moves_white(S,P).
```

Aufgabe 8.6: Backgammon

c), d)

TECHNISCHE UNIVERSITÄT DARMSTADT

c) Definieren Sie ein Prädikat all_possible_moves(P, S), das alle Steine zurückgibt, die gegebener Spieler mit der gegebenen Augenzahl eines Würfels ziehen kann.

```
all_possible_moves(P,S) :- player_stone(P,S), moves(S,X).
```

d) Definieren Sie ein Prädikat beat_opponent (P, S), das alle Steine zurückgibt, die gegebener Spieler P mit den gegeben Augenzahlen (X and\or Y in dice (P, X , Y)) mindestens einen gegnerischen Stein schlagen können.

```
beat_opponent(P,S) :-  player(P,red), all_possible_moves(P,S),
                       moves(S,X), point(T,X), color(T,white).
beat_opponent(P,S) :-  player(P,white), all_possible_moves(P,S
                       moves(S,X),point(T,X), color(T,red).
```

Aufgabe 8.7: Backgammon

a)

TECHNISCHE UNIVERSITÄT DARMSTADT

a) Betrachten Sie das zunächst als ein Datalog-Programm. Bestimmen Sie den Fixpunkt, indem Sie angeben, welche Fakten nach jeder EPP-Iteration hinzukommen.

1. Iteration:

```
fatigue(ana)                fatigue(joe)
headache(john)              body_temerature(ana,38)
body_temerature(joe,37)     body_temerature(john,37)
```

2. Iteration:

```
hypothesis(ana,fever)
```

3. Iteration:

```
hypothesis(ana,flu)
```

Aufgabe 8.7: Datalog vs. Prolog

b) SLD Resultion

TECHNISCHE UNIVERSITÄT DARMSTADT

```
hy(ana,flu) → hy(ana,fever), fa(ana)
              hy(ana,fever) → bt(ana,X), X>37.5
                              bt(ana,X) → Exit:  {X/38}
                              38 > 37.5 → Exit
              fa(ana)       → Exit
```

Aufgabe 8.7: Datalog vs. Prolog

b) SLD Resultion

TECHNISCHE UNIVERSITÄT DARMSTADT

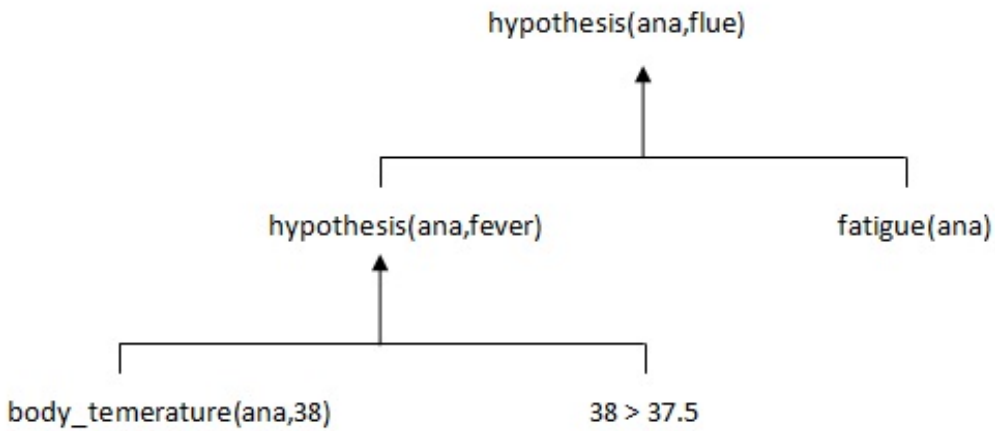
```
hy(john,flu) → hy(ana,fever), fa(john)
               hy(john,fever) → bt(john,X), X>37.5
                                bt(john,X) → Exit:  {X/37}
                                37 > 37.5 → Fail

hy(john,flu) → hy(john,fever), ha(john)
               hy(john,fever) → bt(john,X), X>37.5
                                bt(john,X) → Exit:  {X/37}
                                37 > 37.5 → Fail
```

Aufgabe 8.7: Datalog vs. Prolog

c) Beweisbaum

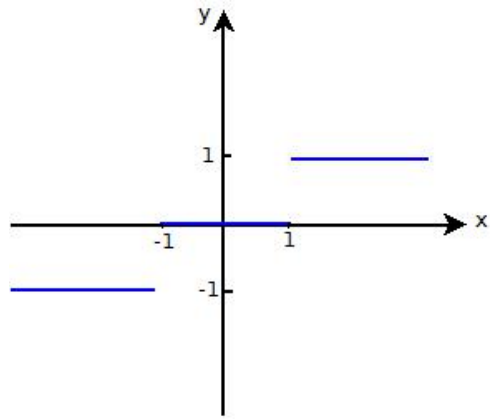
TECHNISCHE UNIVERSITÄT DARMSTADT

c) Wie sieht der Beweisbaum aus?



Aufgabe 8.8: Cut

TECHNISCHE UNIVERSITÄT DARMSTADT

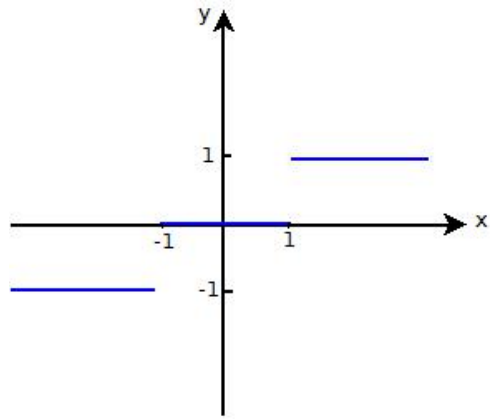



```
f(X,1) :- 1 < X.
f(X,0) :- X < 1.
f(X,-1) :- X < -1.
```

- liefert für Werte kleiner als -1 zwei Werte 0 und -1 zurück.

Aufgabe 8.8: Cut

TECHNISCHE UNIVERSITÄT DARMSTADT

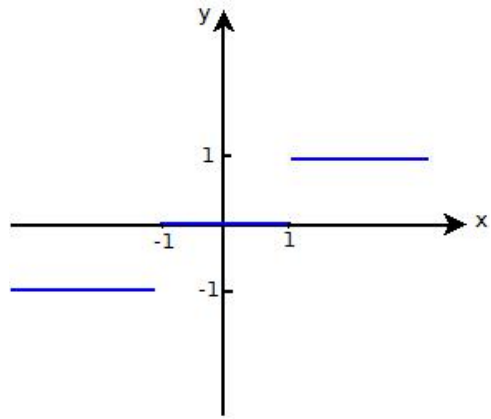



```
f(X,1) :- 1 < X.
f(X,0) :- X < 1, !.
f(X,-1) :- X < -1, !.
```

- immer wenn X < 1, dann wird die Suche nach einer anderen Lösung abgebrochen.

Aufgabe 8.8: Cut

TECHNISCHE UNIVERSITÄT DARMSTADT




```
f(X,1) :- 1 < X, !.
f(X,-1) :- X < -1, !.
f(X,0) :- X < 1, !.
```

Nur das obige Programm beschreibt die gegebene Funktion.

- Falls 1 < X dann wird die Suche abgebrochen Y = 1.
- Falls X < -1 dann ist das einzige Ergebnis Y = -1.
- Falls X < 1 nämlich falls -1 < X < 1 dann Y = 0.