SE3-LP Übungsblatt 01

Arne Beer
Anne-Victoria Meyer

1

1.1

`[familie].`
Der Ausdruck gibt "true" zurück, da das Einbinden der File erfolgreich war. Die Ausdrücke sind vom Effekt her zueinander äquivalent. [] deklariert in in Prolog eine Liste. Mit dem . Operator wird nun jedes Element dieser Liste wie ein Befehl ausgeführt. Wenn man allerdings einen Pfad angibt [/dir/file] sucht er in dem momentenan Directory nach der File /dir/file, welche natuerlich nicht existiert. Daher muss der der Ausdruck in " gesetzt werden, damit er im Pfad sucht.

1.2

`listing.`
Dieser Befehl gibt alle Prädikate, welche in der momentanen Datenbank vorhanden sind, mit Klauseln und Fakten aus.
`listing(mutter_von).`
Gibt alle Prädikate vom Typen mutter_von mit ihren Klauseln und Fakten aus.

1.3

`assert(mutter_von(marie, tom)).`
assert ist deprecated und soll daher nicht benutzt werden. Stattdessen soll assertz benutzt werden.

```
assertz(mutter_von(marie, rom)).
asserta(mutter_von(marie, fom)).
```
asserta sorgt dafür, dass die Klausel am Anfang der Klauseln des entsprechenden Prädikats steht. assertz hingegen hängt die Klausel an das Ende der Klauseln des entsprechenden Prädikats.
```
 listing(mutter_von).
```

2

2.1

a

```
Vater_von(johannes, andrea).
true.
```
Johannes ist also der Vater von Andrea.

b

```
mutter_von(helga, charlotte).
false
```
Nein, die Mutter von Charlotte heisst nicht Helga.

c

```
vater_von(Vater, magdalena).
Vater = walter.
```
Der Vater von Magdalena heisst also Walter. Vater ist hier eine Variable. Die einzige Belegung für die "vater_von(Vater, magdalena)." true ist, ist also wenn Vater=walter.

d

```
vater_von(Vater, walter).
false
```

Es gibt keine Möglichkeit für den Ausdruck, wahr zu werden. Laut Datenbank hat Walter also keinen Vater.

e

```
vater_von(otto, Kind).
Kind = hans ;
Kind = helga.
```

Ottos Kinder heissen Hans und Helga.

f

```
vater_von(V, K).
V = otto,
K = hans ;
V = otto,
K = helga ;
V = gerd,
K = otto ;
V = johannes,
K = klaus ;
V = johannes,
K = andrea ;
V = walter,
K = barbara ;
V = walter,
K = magdalena.
mutter_von(M, K).
M = marie,
K = hans ;
M = marie,
K = helga ;
M = julia,
K = otto ;
```

```
M = barbara,
K = klaus ;
M = barbara,
K = andrea ;
M = charlotte,
K = barbara ;
M = charlotte,
K = magdalena.
```

Alle möglichen Belegungen (die true liefern würden) werden angezeigt. So ist z.B. Otto der Vater von Hans:

```
V = otto,
K = hans ;
```

Semikola separieren die unterschiedlichen Belegungen wobei sie oder-Verknüpfungen sind. Die Kommata separieren die jeweiligen Variablenwerte, sind also und-Verknüpfungen.

g

```
\+ vater_von(klaus, Kind).
true.
```

Klaus hat also keine Kinder. Es gibt keine wahre Belegungen für diese Klausel, daher würde false ausgegeben werden. Das wird hier jedoch noch negiert also erhalten wir true.

h

```
\+ vater_von(otto, Kind).
```

`false.` Otto hat also Kinder. Auch wenn die Klausel vater_von(otto, Kind). die wahren Belegungen zurück geben würde, wird bei der Negation lediglich ein boolscher Wert ausgegeben.

i

```
\+ \+ vater_von(otto, Kind).
true.
```

Durch die Negation erhalten wir einen Wahrheitswert, durch doppelte den gewünschten.

2.2

```
mutter_von(charlotte, Kind), (mutter_von(Kind, EnkelKind1);
vater_von(Kind, EnkelKind2)).
Kind = barbara,
EnkelKind1 = klaus ;
Kind = barbara,
EnkelKind1 = andrea ;
false.
```

Wir suchen eine Belegung für das Kind von Charlotte, so dass dieses Kind ebenfalls eines hat. Da es kein kind_von Praediakt gibt müssen vater_von und mutter_von verodert werden. Wir haben die Klauseln geklammert, um die Operationen in der richtigen Reihenfolge durchzuführen. Die logischen Verknüpfungen sind die folgenden:
; = oder
, = und
Am Ende der Ausgabe steht ein false, da zunächst die beiden möglichen Belegungen für eine wahre Auswertung aufgelistet werden. Und dann alle anderen, die alle auf false auswerten.

2.3

Anders als im Skript werden nur die erfolgreichen Belegungen auch tatsächlich aufgelistet, es sei denn es gibt keine erfolgreichen.

```
vater_von(Vater, walter).
Call: (6) vater\_von(\_G1814, walter) ? creep
Fail: (6) vater-\_von(\_G1814, walter) ? creep
false.
```

In dieser trace ist also keine wahre Belegung gefunden worden.

```
vater_von(otto, Kind).
Call: (6) vater\_von(otto, \_G1815) ? creep
```

```
Exit: (6) vater\_von(otto, hans) ? creep
Kind = hans ;
Redo: (6) vater\_von(otto, \_G1815) ? creep
Exit: (6) vater\_von(otto, helga) ? creep
Kind = helga.
```

In dieser trace ist eine wahre Belegung gefunden worden, danach wird jedoch weiter probiert. Dabei wird offensichtlich ähnlich wie in der Vorlesung nach einem Baumschema vorgegangen. Die Misserfolge werden nicht angezeigt.