Miriam Butt und Maribel Romero
G220/5109, G212/2728
{miriam.butt|maribel.romero}@uni-konstanz.de

Advanced Computational Semantics
Sommersemester 2008, Hauptseminar

Lösung zu Übung 6, 05.06.2008

Aufgabe 1)

1. (a) *a woman shooots a man* funktioniert:

```
?- s([a,woman,shoots,a,man],[]).
true ;
fail.
```

(b) Der Befehl `listing.` zeigt den eigentlichen Aufbau der Prolog-Grammatik. Die Schreibweise der Phrasenstrukturregeln mit den Pfeilen ist im Prinzip nur ein optischer Umweg um die etwas komplizierteren *Difference Lists*, die von Prolog für Grammatiken verwendet werden.

(c) Man übergibt dem Prädikat `s/2` einfach eine beliebige Variable als erstes und wie oben eine leere Liste als zweites Argument:

```
?- s(X,[]).
X = [the, woman, shoots, the, woman] ;
X = [the, woman, shoots, the, man] ; ...
```

Die Grammatik umfasst insgesamt 20 Sätze.

2. Für die Sätze (a) bis (c) mussten nur neue Lexikoneinträge erstellt werden. (d) allerdings erfordert eine komplexere NP-Regel mit Rekursion, was sich nicht ohne Umbenennen der Kategorien erreichen lässt:[1]

[1] Siehe dazu *Learn Prolog Now!*, v.a. Seiten 131-133.

```
s --> np,vp.
np --> npsimple.
np --> npsimple, conj, np.
npsimple --> det,n.
vp --> v,np.
vp --> v.
det --> [the].
det --> [a].
n --> [woman].
n --> [man].
n --> [apple].
n --> [cat].
n --> [dog].
v --> [shoots].
v --> [laughs].
v --> [chases].
conj --> [and].
```

3. *a woman and a man shoots the apple* und andere Sätze dieser Art werden deswegen von der Grammatik als korrekt erkannt, weil sie nichts über die Kongruenz im Englischen weiß (also darüber, dass sich das Verb in Numerus und Person nach dem Subjekt richtet).

Aufgabe 2)

1. Zum Beispiel so:

```
?- s(Sem,[vincent,walks],[]).
Sem = walk(vincent) ;
fail.

?- s(Sem,[a,woman,walks],[]).
Sem = some(_G255, and(woman(_G255), walk(_G255))) ;
fail.

?- s(Sem,[every,woman,likes,a,foot,massage],[]).
Sem = all(_G273, imp(woman(_G273), some(_G282, and(footmassage(_G282),
```

```
like(_G273, _G282))))) ;
fail.
```

2. Folgende Lexikoneinträge hat man für die Sätze in (a) bis (d) einfügen müssen:

Satz (a): `iv(Y,laugh(Y))--> [laughs].`
Satz (b): `tv(Y,Z,shoot(Y,Z))--> [shoots].`
Satz (b): `det(X,Restr,Scope,some(X,and(Restr,Scope)))--> [the].`
Satz (c): `tv(Y,Z,chase(Y,Z))--> [chases].`
Satz (d): `noun(X,man(X))--> [man].`

Aufgabe 3)

1. Diejenigen Beispiele, die mehrdeutig sind laut Prolog, besitzen >1 `readings`.
Hier die ambigen Beispielsätze:

Satz:	Ambig, weil:
1) [every, woman, or, a, man, dances] (2 readings)	Skopus von *a, every*
2) [every, customer, in, a, restaurant, smokes] (2 readings)	Skopus von *a, every*
3) [mia, or, vincent, eats, a, quarter, pounder, with, cheese] (2 readings)	Skopus von *a*
4) [every, customer, drinks, a, five, dollar, shake] (2 readings)	Skopus von *a*
5) [a, customer, knows, mia, or, a, man] (2 readings)	Skopus von *a*
6) [vincent, knows, every, woman, or, a, man] (2 readings)	Skopus von *a, every*
7) [a, robber, likes, every, customer, in, a, restaurant] (5 readings)	2 x Skopus von *a*, Skopus von *every*
8) [butch, kills, a, criminal, or, shoots, a, criminal] (2 readings)	Skopus von *a*

9) [a, boxer, does, not, die] (2 readings)	Skopus von *a*
10) [every, boxer, does, not, die] (2 readings)	???
11) [a, man, does, not, smoke, or, dance] (2 readings)	Skopus von *a*
12) [every, customer, in, a, restaurant, eats, a, big, kahuna, burger] (5 readings)	2 x Skopus von *a*, Skopus von *every*
13) [every, customer, in, a, restaurant, does, not, eat, a, big, kahuna, burger] (18 readings)	2 x Skopus von *a*, Skopus von *every*
14) [every, man, in, a, restaurant, knows, a, woman, with, a, car] (14 readings)	3 x Skopus von *a*, Skopus von *every*
15) [if, every, man, knows, a, woman, then, every, woman, knows, a, man] (4 readings)	2 x Skopus von *a*, Skopus von *every*

Anmerkungen: Die Ausgabe von Prolog ist sehr inkonsistent, was die Ambiguität der Sätze angeht. Es wird nicht klar, warum Satz 10) ambig sein sollte. Des Weiteren liegt bei Satz 13) eine Kombination der Skopi von *a* (2 x) und *every* (1 x) vor; die Zahl der Lesarten ist jedoch höher als in 14), wo noch mehr Quantoren enthalten sind.
Es ging bei dieser Aufgabe hauptsächlich darum, die Testsuite durchlaufen zu lassen und die Quantoren als Ursache der Ambiguitäten zu identifizieren.

2. Die Grammatik zu `lambdaDRT.pl` befindet sich in `englishGrammar.pl`, das Lexikon in `englishLexicon.pl`. Hier musste also nachgeschaut werden, mit was für Sätzen man das Programm füttern kann. Hier eine Beispielausgabe:

```
?- lambdaDRT.
> A woman dances. Vincent collapses. Every boxer collapses.
1 drs([A, B, C, D], [pred(woman, A), pred(dance, B), rel(agent, B, A),
pred(event, B), pred(vincent, C), pred(collapse, D), rel(agent, D, C),
pred(event, D), imp(drs([E], [pred(boxer, E)]), drs([F], [pred(collapse, F),
rel(agent, F, E), pred(event, F)]))])
true ;
fail.
```

Aufgabe 3)

Es mussten nur die beiden Lexikoneinträge für *laugh* und *kiss* bei `englishLexicon.pl` eingefügt werden.

a)
```
lexEntry(iv,[symbol:laugh,syntax:[laugh],inf:inf,num:sg]).
lexEntry(iv,[symbol:laugh,syntax:[laughs],inf:fin,num:sg]).
lexEntry(iv,[symbol:laugh,syntax:[laugh],inf:fin,num:pl]).
```

b)
```
lexEntry(tv,[symbol:kiss,syntax:[kiss],inf:inf,num:sg]).
lexEntry(tv,[symbol:kiss,syntax:[kisses],inf:fin,num:sg]).
lexEntry(tv,[symbol:kiss,syntax:[kiss],inf:fin,num:pl]).
```