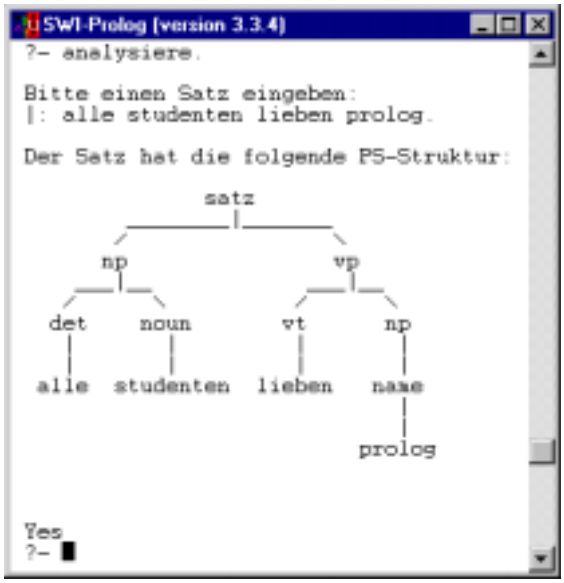
Kapitel 7.
Eine Phrasenstrukturgrammatik mit Prolog

Wie am Ende des vorherigen Kapitels bereits gesagt wurde, ist der um die Ersetzungs-PS-Regeln erweiterte Pattern-Matcher-Akzeptor zweifellos in der Lage, bezüglich derjenigen Komponenten modifiziert zu werden, die das Ganze aus linguistischer Perspektive interessant machen. Das Problem besteht dabei im Aufbau des Programmes: die Erweiterung dahingehend, dass Sätze nicht nur akzeptiert werden, sondern für einen jeden Satz auch eine bzw. mehrere Strukturbeschreibungen generiert werden, ist zwar machbar, aber ein wenig kniffelig. Diese Komplikation umgehen wir, indem wir die Implementierung der Phrasenstruktur von vorneherein anders anlegen als den Patter-Matcher.

In den nächsten Abschnitten wird es darum gehen, schrittweise ein Programm zu entwickeln, welches die folgenden Leistungen aufweist:

1. Es kann entscheiden, ob ein englischer Satz syntaktisch wohlgeformt ist
2. Es kann einem wohlgeformten Satz eine entsprechende Strukturbeschreibung zuordnen.
3. Bei struktureller Ambiguität eines Satzes wird für jede Interpretation eine Strukturbeschreibung geliefert.

Der nachstehende Screen-Shot zeigt, wie das Ganze letztendlich aussehen soll:



Natürlich kann dieses Programm nicht umfassend und erschöpfend sein dahingehend, dass es für alle erdenkbaren Sätze diese Aufgaben erfüllt. Es soll vielmehr eine erste Vorstellung davon erzeugen, welche Problemstellungen in diesem Bereich der maschinellen Sprachverarbeitung auftreten und wie man bestimmte Erkenntnisse aus dem Bereich der Syntaxtheorie zweckdienlich in einem Computerprogramm umsetzen kann. Das Programm ist für die englische Sprache ausgelegt, weil in dieser bestimmte Unannehmlichkeiten, die eine Analyse erschweren, gar nicht erst auftreten (wie z.B. die Flexion der Artikel und Adjektive oder die diskontinuierlichen Elemente im Verbalsyntagma). Ansatzsweise aber werden – wie im Pattern-Matcher auch – diese Phänomene ebenfalls angegangen.

Wir werden zunächst kleine Brötchen backen und Punkt 1. der oa. Aufgabenstellungen bearbeiten, d.h. einen auf einer Phrasenstrukturgrammatik basierenden Akzeptor realisieren, um das Programm dann sukzessive um die anderen Punkte (zu denen neben der Generierung einer Strukturbeschreibung auch die Art und Weise, in der zu analysierende Wortketten eingegeben werden, gehört) zu erweitern und verbessern.

7.1. Teil 1: Ein PSG-Akzeptor

An dem folgenden Beispielsatz soll die Grundlage für das Analyseprogramm erklärt werden:

7.1. *the dog chased the cat*

Ein wesentlicher Aspekt der menschlichen Sprachverarbeitung (der allerdings so unbewußt abläuft, dass man ihn sich erstmal bewußt machen muß) ist die Tatsache, dass eine solche Wortfolge völlig automatisch linear abgearbeitet wird – was die englische geschriebene Sprache betrifft von links nach rechts. Es ist also ganz klar, dass zuerst das Wort *the* kommt, dass *chased* zwischen *the* und *dog* steht, dass die Wortfolge mit *cat* endet usw. Die Reihenfolge der einzelnen Wörter ist natürlich, wie wir bereits im Abschnitt über den Pattern-Matcher gesehen haben, ein zentrales Kriterium bei der Analyse eines Satzes. Was damit gesagt werden soll ist, dass die lineare Reihenfolge der Wörter wie eben in *the dog chased the cat* eine der implizit im Satz enthaltenen Informationen ist, die für Prolog explizit gemacht werden muß (ähnliche Fragestellungen tauchten ja bereits im Stammbaum und im semantischen Netz auf).

Dazu bietet es sich an, den Satz wie folgt zuerst einmal in einzelne Segmente zu zerteilen:

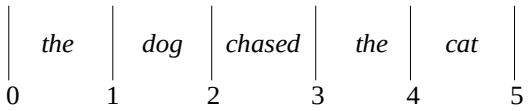


Abb. 7.1

Die Numerierung der einzelnen Positionsstriche ermöglicht, über den Satz *the dog chased the cat* folgende Aussage zu machen:

Der Satz the *dog chased the cat* besteht aus fünf Segmenten, wobei die Anfangsposition 0 mit der Position 1 durch das Wort *the* verbunden ist, die Position 1 mit der Position 2 durch das Wort *dog*, die Position 2 mit der Position 3 durch das Wort *chased*, die Position 3 mit der Position 4 durch das Wort *the* und die Position 4 mit der Endposition 5 durch das Wort *cat*.

Die Kette der einzelnen Wörter als Verbindung jeweils zweier Positionen kann wie folgt graphisch illustriert werden:

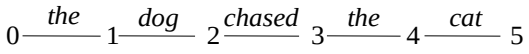


Abb. 7.2.

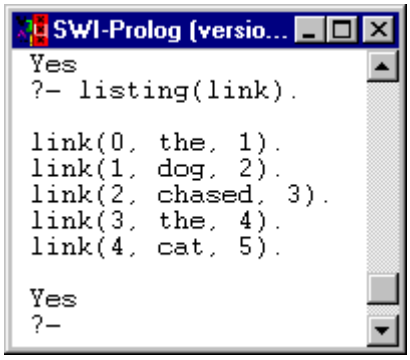
Es stellt sich zunächst die Frage, wie eine solcher Satz in einer für die Weiterverarbeitung durch Prolog geeigneten Form repräsentiert werden kann. Für den Benutzer das einfachste Verfahren wäre es, wie im Pattern-Matcher Sätze als Listen über die Tastatur einzugeben, und den Rest Prolog zu überlassen. Genau so wollen wir verfahren. Im Augenblick aber beschränken wir uns erstmal darauf, diese Information in Form von Fakten einzugeben, und zwar mithilfe eines Prädikats link/3, bei welchem das erste Argument die Ausgangsposition ist, das zweite Argument das verbindende Wort und das dritte Argument die Zielposition. Das Wort *the* stellt die Verbindung her zwischen den Positionen 0 und 1, also gilt als Prolog-Fakt:

```
link(0,the,1).
```

Das Wort *dog* stellt die Verbindung her zwischen den Positionen 1 und 2, ergo

```
link(1,dog,2).
```

und so weiter. Mit den uns bis jetzt zur Verfügung stehenden Mitteln müssen wir erstmal **jeden** Satz, den wir analysieren wollen, auf diese Art eingeben.[29] Dieses Verfahren werden wir weiter unten allerdings automatisieren. Für die Weiterverbeitung (sprich Analyse) muss der Satz aus Beispiel 7.1. wie folgt in der Prolog-Wissensbasis repräsentiert sein:



Hier mag man sich fragen – wieso so umständlich? Wenn wir die zu analysierende Wortkette in Listenform eingeben, also analog zur Herangehensweise im Pattern-Matcher, wird deren Linearität doch viel direkter abgebildet, als es hier der Fall ist. So wir von jedem im zu analysierenden Satz vorkommenden Wort wissen, zu welcher Wortart es gehört (und diese Information ist im Lexikon, welches wir auch hier verwenden, in Form von lex/2-Fakten notiert) sind wir jetzt in der Tat nicht weiter als im Pattern-Matcher (ohne Satzmuster). Es soll an dieser Stelle auch nicht verheimlicht werden, dass das ganze anstehende Verfahren extrem vereinfacht werden könnte, wenn man den in Prolog eingebauten **D**(efinite) - **C**(lause) - **G**(rammar)-Formalismus verwendet. In diesem Formalismus, auf den wir später ggf. näher eingehen werden und der am Ende des Kapitels genauer vorgestellt wird, sind ganz bestimmte Verfahrensschritte bei der Umsetzung einer Phrasenstrukturgrammatik in Prolog bereits

[29]Das bedeutet, dass ein Satz wie *the teacher wrote a book* ebenfalls in 'mundgerechten Prolog–Happen' eingegeben würde:
link(0,the,1). link(3,a,4).
link(1,teacher,2). link(4,book,5).
link(2,wrote,3).

vordefiniert, so dass man sich bei der Implementierung über bestimmte Dinge – wie eben 'in welcher Form muss die Wortkette in der Wissensbasis repräsentiert sein' überhaupt keine Gedanken mehr machen muss. Da es aber ein Ziel der Veranstaltung ist, über die Vermittlung einiger zentraler Prolog-Grundlagen die Teilenehmenden in die Lage zu versetzen, bestimmte Problemstellungen mehr oder weniger 'from scratch' selbständig angehen zu können, gehen wir zunächst den etwas umständlicheren, dafür aber in seinen einzelnen Schritten völlig transparenten Weg. Auf dieser Basis lassen sich dann vordefinierte Programmschemata wie der DCG-Formalismus auch besser nachvollziehen. Damit zurück zum Problem.

Die Kombination der link/3 Fakten und des Lexikons liefert uns sozusagen die lineare Ebene des Satzes. In der PSG geht es aber ja um mehr – nämlich auch um die hierarchische Ebene der syntaktischen Kategorien. An dieser Stelle wird deutlich, wieso wir die link/3-Notation für die zu analysierende Wortkette verwenden: um nämlich die syntaktischen Kategorien für den gegebenen Satz darzustellen, haben wir durch die Einteilung des Satzes in einzelne Segmente bereits eine vorzügliche Basis geschaffen. Die Strukturbeschreibung von *the dog chased the cat* hätte (im Rahmen der traditionellen Konstituentenanalyse) auf dieser Basis die folgende Form:

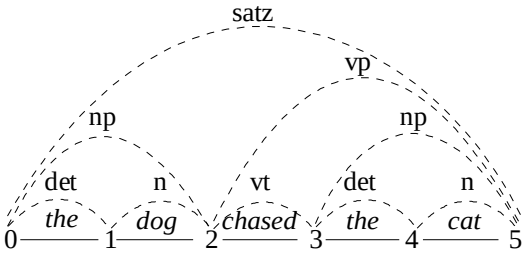


Abb. 7.3.

Nun können die folgenden Aussagen getroffen werden:

1. Der Bogen, der die Positionen 0 – 5 umspannt, entspricht einem Satz. Die Position 2 zerlegt diese Spanne derart, dass der Bogen von 0 – 2 einer Nominalphrase entspricht und der Bogen von Position 2 – 5 einer Verbalphrase.
2. Der Bogen, der die Positionen 0 – 2 umspannt, entspricht einer Nominalphrase. Position 1 zerlegt die Spanne von 0 – 2 derart, dass der Bogen von 0 – 1 einem Determinator entspricht und der Bogen von 1 – 2 einem Nomen.
3. Der Bogen, der die Positionen 2 – 5 umspannt, entspricht einer Verbalphrase. Position 3 zerlegt die Spanne von 2 – 5 derart, dass der Bogen von 2 – 3 einem transitiven Verb entspricht und der Bogen von 3 – 5 einer Nominalphrase.
4. Der Bogen, der die Positionen 3 – 5 umspannt, entspricht einer Nominalphrase. Position 4 zerlegt die Spanne 3 – 5 derart, dass der Bogen von 3 – 4 einem Determinator entspricht und der Bogen von 4 – 5 einem Nomen.

Zunächst betrachten wir die Aussage unter Punkt 1:

> Der Bogen, der die Positionen 0 – 5 umspannt, entspricht einem Satz. Die Position 2 zerlegt diese Spanne derart, dass der Bogen von 0 – 2 einer Nominalphrase entspricht und der Bogen von Position 2 – 5 einer Verbalphrase.

Diese Aussage, die sich auf den konkreten Satz und also die konkreten Positionen 0, 2 und 5 bezieht, kann auf die folgende Art verallgemeinert und paraphrasiert werden:

Ein Bogen, der von einer Position P0 zu einer Position Pn reicht, ist dann ein Satz, wenn zwischen P0 und Pn eine weitere Position P1 liegt, welche die Spanne P0 – Pn so zerlegt, dass P0 – P1 eine Nominalphrase ist und P1 – Pn eine Verbalphrase.

Dieser verallgemeinerte Ausdruck kann direkt in eine Prolog-Regel übersetzt werden:

```
satz(P0,Pn):-
        np(P0,P1),
        vp(P1,Pn).
```

Dabei sind P0, P1, Pn Variablen, die Werte für unterschiedliche Positionen annehmen können. Auf die gleiche Weise verfahren wir mit den anderen Bögen:

> Der Bogen, der die Positionen 0 – 2 umspannt, entspricht einer Nominalphrase. Position 1 zerlegt die Spanne von 0 – 2 derart, das der Bogen von 0 – 1 einem Determinator entspricht und der Bogen von 1 – 2 einem Nomen.

Auch diese Aussage wird verallgemeinert paraphrasiert:

Ein Bogen, der von einer Position P0 zu einer Position Pn reicht, ist dann eine Nominalphrase, falls zwischen P0 und Pn eine weitere Position P1 liegt, welche die Spanne P0 – Pn so zerlegt, dass P0 – P1 ein Determinator ist und P1 – Pn ein Nomen.

Als Regel:

```
np(P0,Pn):-
        det(P0,P1),
        noun(P1,Pn).
```

An dieser Stelle schon sieht man den Vorteil der Verallgemeinerung: mit dieser Regel für die Nominalphrase ist nicht nur der Bogen, der in Abbildung **7.3.** die Positionen 0 – 2 umspannt (*the dog*) erfasst, sondern auch der Bogen, der die Positionen 3 – 5 umspannt (*the cat*).

Nun fehlt nur noch die Regel für den Verbalphrasenbogen:

> Der Bogen, der die Positionen 2 – 5 umfaßt, entspricht einer Verbalphrase. Position 3 zerlegt die Spanne von 2 – 5 derart, dass der Bogen von 2 – 3 einem transitiven Verb entspricht und der Bogen von 3 – 5 einer Nominalphrase.

Verallgemeinert:

Ein Bogen, der von einer Position P0 zu einer Position Pn reicht, ist dann eine Verbalphrase, wenn zwischen P0 und Pn eine weitere Position P1 liegt, welche die Spanne P0 – Pn so zerlegt, dass P0 – P1 ein transitives Verb ist und P1 – Pn eine Nominalphrase.

Als Regel:

```
vp(P0,Pn):-
        vt(P0,P1),
        np(P1,Pn).
```

Die Regeln auf der phrasalen Ebene sind somit für den Satz, die Nominalphrase und die Verbalphrase eingegeben. Was noch fehlt, sind Regeln für die einzelnen lexikalischen Kategorien Determinator, Nomen, und Verb. Dazu betrachten wir die folgenden, wiederum auf Abbildung **7.3.** bezogenen Feststellungen:

1. Die Positionen 0 – 1 werden durch das Wort *the* verbunden, welches der lexikalischen Kategorie DETERMINATOR angehört.
2. Die Positionen 1 – 2 werden durch das Wort *dog* verbunden, welches der lexikalischen Kategorie NOMEN angehört.
3. Die Positionen 2 – 3 werden durch das Wort *chased* verbunden, welches der lexikalischen Kategorie TRANSITIVES VERB angehört.
4. Die Positionen 3 – 4 werden durch das Wort *the* verbunden, welches der lexikalischen Kategorie DETERMINATOR angehört.
5. Die Positionen 4 – 5 werden durch das Wort *cat* verbunden, welches der lexikalischen Kategorie NOMEN angehört.

Auch hier verallgemeinern wir und paraphrasieren die erste Aussage wie folgt:

Ein Bogen, der von einer Position P0 zu einer Position Pn reicht ist dann ein Determinator, wenn P0 und P1 durch ein Wort verbunden sind, welches der lexikalischen Kategorie Determinator angehört.

Diese Aussage wird wie folgt in Prolog übersetzt:

```
det(P0,P1):-
        link(P0,Wort,P1),
        lex(Wort,determinator).
```

Auf das Lexikon bezogen findet sich wg. Schreibfaulheit für die letzte Zielklausel lex(Wort,determinator) wahrscheinlich nur ein einziger Eintrag: Wort = the. Aber dennoch wird auch hier der Vorteil der Generalisierung deutlich, schließlich kann das Lexikon durch eine beliebige Zahl von Einträgen erweitert werden, auf welche die Regel für den Determinator dann angewendet werden kann:

```
lex(a,determinator).
lex(this,determinator).
lex(my,determinator). usw.
```

Nun können wir, analog zur Regel für den Determinator, die Regeln für die anderen lexikalischen Kategorien aufführen:

```
noun(P0,Pn):-
        link(P0,Wort,P1),
        lex(Wort,noun).
vt(P0,Pn):-
        link(P0,Wort,P1),
        lex(Wort,vt).
```

Wenn diese Daten in Prolog eingegeben sind und das Lexikon konsultiert wurde, enthält die Wissensbasis

- die Regeln für den Satz und die phrasalen Kategorien NP und VP
- die Regeln für die lexikalischen Kategorien N, Det und Vt,
- den zu analysierenden Satz *the dog chased the cat* in Form von link/3 Fakten

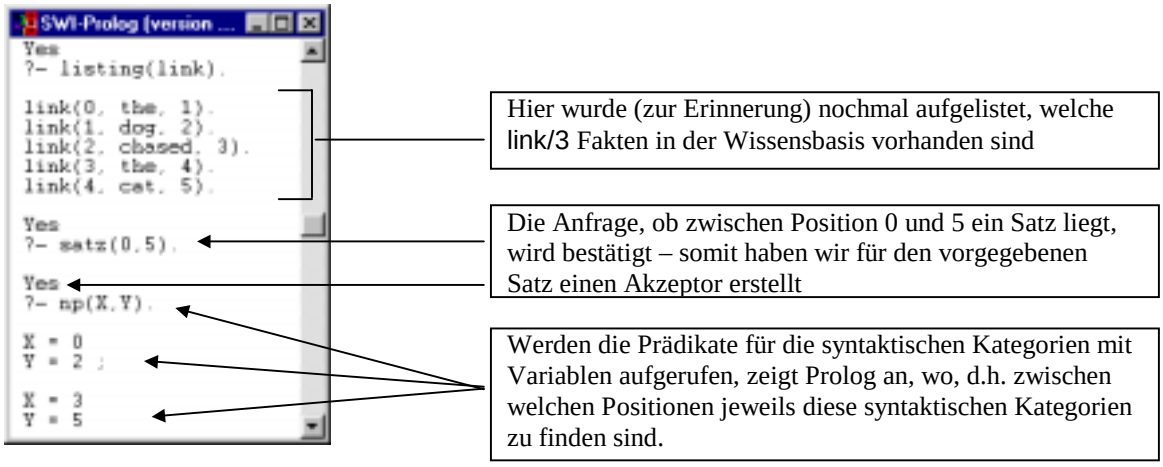
An dieser Stelle können bestimmte Anfragen an das Programm gestellt werden. Beispielsweise können wir überprüfen, ob zwischen den Positionen 0 und 5 ein Satz liegt – ob also die als verbindet/3 Fakten eingegebene Wortkette ein Satz ist:

Liegt zwischen Position 0 und Position 5 ein (wohlgeformter) Satz?
Eingabe: satz(0,5) Ausgabe: yes

Wir können aber, durch die Verwendung von Variablen, auch anzeigen lassen, zwischen jeweils welchen Positionen eine NP, eine VP oder ein Satz liegt:

Zwischen welchen Positionen liegt eine Nominalphrase?
Eingabe: np(X,Y) Ausgabe: X = 0, Y = 2
X = 3, Y = 5

Das Ganze sieht 'in echt' so aus:



Damit wäre das erste Ziel erreicht – wir haben auf der Basis einer PSG einen Aktzeptor geschrieben, der für eine englische Worfolge (in unserem Fall allerdings, aufgrund der schmalen Datenlage, leider nur die

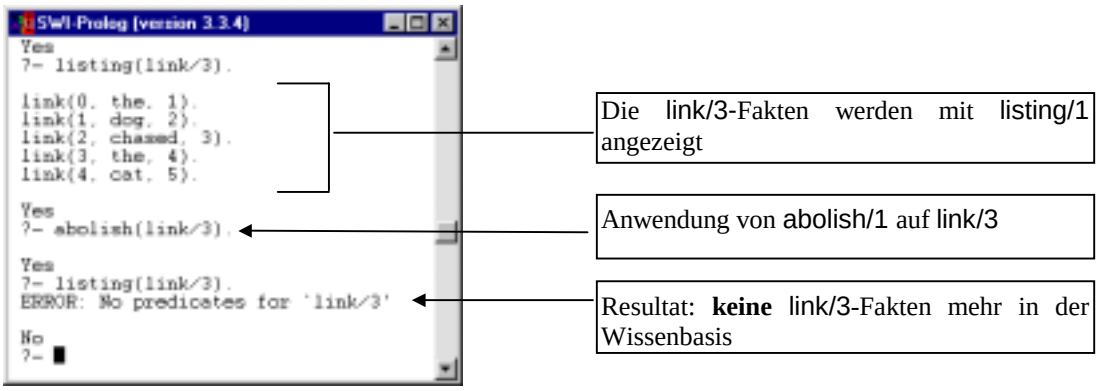
eine Wortfolge und nur für Sätze mit der Struktur ((Det Noun)$_{NP}$ (Vt (Det Noun)$_{NP}$)$_{VP}$)$_{S}$) erkennt, ob diese ein Satz ist oder nicht.

Mit Bezug auf eine PSG mit traditioneller Schreibweise umfaßt die Grammatik die folgenden Regeln:

PSG	Prolog-Regeln	
S → NP VP	s(P0,Pn):- np(P0,P1), vp(P1,Pn).	Regeln für die syntaktischen Kategorien
NP → Det N	np(P0,Pn):- det(P0,P1), noun(P1,Pn).	
VP → Vt NP	vp(P0,Pn):- vt(P0,P1), np(P1,Pn).	
	noun(P0,Pn):- link(P0,Wort,Pn), lex(Wort, noun). det(P0,Pn):- link(P0,Wort,Pn), lex(Wort, det). vt(P0,Pn):- link(P0,Wort,Pn), lex(Wort, vt).	Regeln für die lexikalischen Kategorien
Det → *the*	lex(the,det).	Lexikon
N → {*dog, cat*}	lex(dog,noun). lex(cat,noun).	
Vt→ *chased*	lex(chased,vt).	

Um weitere Sätze zu überprüfen, muss folgendes geschehen:

- Zuerst müssen die alten link/3-Fakten aus der Wissensbasis entfernt werden. Dafür verwendet man das Systemprädikat abolish/1, Das Argument von abolish/1 ist eine Angabe über Funktor und Stelligkeit des zu entfernenden Prädikates. Um also alle link/3-Fakten aus der Wissensbasis zu entfernen, muss folgender Befehl eingegeben werden: abolish(link/3)



- Der neue Satz muss in Form von link/3 Fakten eingegeben werden

In den nachfolgenden Abschnitten werden wir dieses Programm so erweitern, dass eine Strukturbeschreibung erzeugt wird, und die Benutzerfreundlichkeit steigern, in dem wir die Eingabe der zu analysierenden Sätze und deren Notation als link/3-Fakten automatisieren.

Aufgabe 1.

Legt eine Datei namens psg1.pl an.
Lasst von dort aus lexikon.pl aufrufen.
Setzt die Grammatik der vorherigen Seite psg1.pl um

Aufgabe 2.

Ergänzt die Datei psg1.pl um Phrasenstrukturregeln, die auch die Analyse der folgenden Sätze ermöglichen und überprüft anschließend, ob die Sätze akzeptiert werden.:
Bart slept
The dog bit Lisa
Barney sold the beer to Homer
Marge read a rather silly cartoon.
Tip: Überlegt Euch zunächst, welche PS-Regeln für diese Sätze nötig sind, und übersetzt diese dann anschließend in Prolog. In den Fällen, in welchen neue Wortarten auftreten (Eigennamen, Präpositionen usw.), müssen natürlich auch entsprechende Regeln für die lexikalischen Kategorien formuliert werden! Wenn die Analyse nicht klappt, kann es auch immer am Lexikon liegen