Parsen mit PC-PATR (*Parsing and Translating*) – CIP-LABOR 25. 07. 08

Einleitung

Der Text einer PATR-Grammatik ist vergleichbar mit dem Quellkode eines Computer-Programmes: es drückt die Regeln der Grammatik in einer Form aus, die von PATR interpretiert bzw. übersetzt wird dahingehend, dass einer gegebenen Eingabekette eine Strukturbeschreibung zugeordnet wird.

Somit ist der PATR-Formalismus quasi eine Programmiersprache für eine maschinelle syntaktische Analyse. Für diese Sprache gelten strenge syntaktische Regeln, die beachtet werden müssen, damit das Programm korrekt arbeiten kann.

Dieser Text führt Sie in die Arbeit mit PATR ein. Sie werden dabei sukzessive eine Mini-Grammatik aufbauen, über die nicht nur der Umgang mit PATR geübt wird, sondern auch einige der zentralen Methoden, Erkenntnisse und Termini der modernen Sprachwissenschaft exemplarisch vermittelt werden.

PATR-Grammatiken

Eine PATR-Grammatik besteht aus einer Menge von Phrasenstruktur-Regeln (PS-Regeln) und einer Menge von Lexikonregeln.

PS-Regeln in PC-PATR

Wie Sie feststellen werden, können PS-Regeln der Form S → NP VP oder AP → A geradezu direkt in PATR ausgedrückt werden. Es erfolgen aber noch einige zusätzliche Angaben: beispielsweise hat jede Regel einen spezifisch ausgedrückten Namen; jede Regel muss mit einem Punkt abgeschlossen werden. Im Einzelnen ist eine PS-Regel in PC-PATR aus folgenden Teilen (in der angegebenen Reihenfolge) aufgebaut:

1. dem Schlüsselwort *Rule*
2. einem REGELNAMEN, der in geschweifte Klammern eingeschlossen wird ({ })
3. einer Phrasenstruktur-Regel aus folgenden Elementen:
   a) dem zu expandierenden nicht-terminalen Symbol als **KOPF**
   b) einem Pfeil (–>) oder Gleichheitszeichen (=)
   c) dem **RUMPF**, der entweder leer ist oder aus einem oder mehreren terminalen oder nicht-terminalen Symbolen[1] besteht, die ggf. als alternative (Schweifklammern) oder optionale (runde Klammern) Konstituenten markiert sind
4. einem Punkt (.)

Der nachstehende annotierte Screenshot zeigt den Aufbau einer einfachen PATR-PS-Regel:

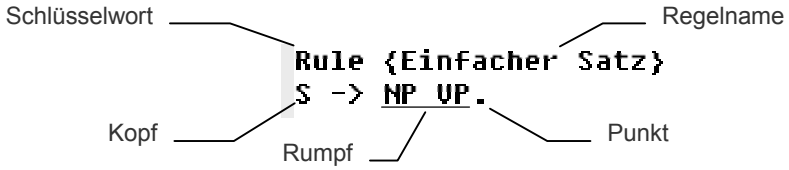


Abb. 1: PS-Regel in PATR

Lexikonregeln in PC-PATR

Damit mit unserer Grammatik Sätze analysiert werden können, brauchen wir ein Lexikon, das die in den zu analysierenden Daten vorkommenden Wortformen und deren Eigenschaften enthält. Das Lexikon besteht aus einer Folge von Lexikoneinträgen, die in der einfachsten Form folgende Gestalt haben:

\w	*Wortform*	z.B.	\w	Mary
\c	*Kategorie*		\c	Name

Der nachstehende Screenshot zeigt die Lexikoneinträge für die Wörter *Mary*, *John*, *laughed* und *jumped*:

```
1   \w Mary
2   \c Name
3
4   \w John
5   \c Name
6
7   \w laughed
8   \c V
9
10  \w jumped
11  \c V
```

Abb. 2: Lexikoneinträge in PATR

[1] Terminale Symbole können nicht weiter expandiert werden und daher nur im Regelrumpf stehen.

Der Editor

PS-Regeln und Lexikonregeln werden in verschiedenen Dateien abgespeichert. Dateien, die die PS-Regeln enthalten, müssen, um von PATR korrekt umgesetzt zu werden, die Dateikennung **.grm** aufweisen; Dateien mit Lexikonregeln erhalten die Kennung **.lex**. An diese Vorgabe müssen Sie sich halten, da nur entsprechend benannte Dateien von PATR interpretiert werden können.

Wir schreiben diese Dateien aber nicht in PATR, sondern in einem Texteditor. Für diesen Zweck ist jeder Editor geeignet; im CIP-Labor arbeiten wir mit dem Editor MED, der PATR als Default-Editor zugeordnet ist.

Die Arbeit mit PATR kann schematisch wie folgt dargestellt werden:

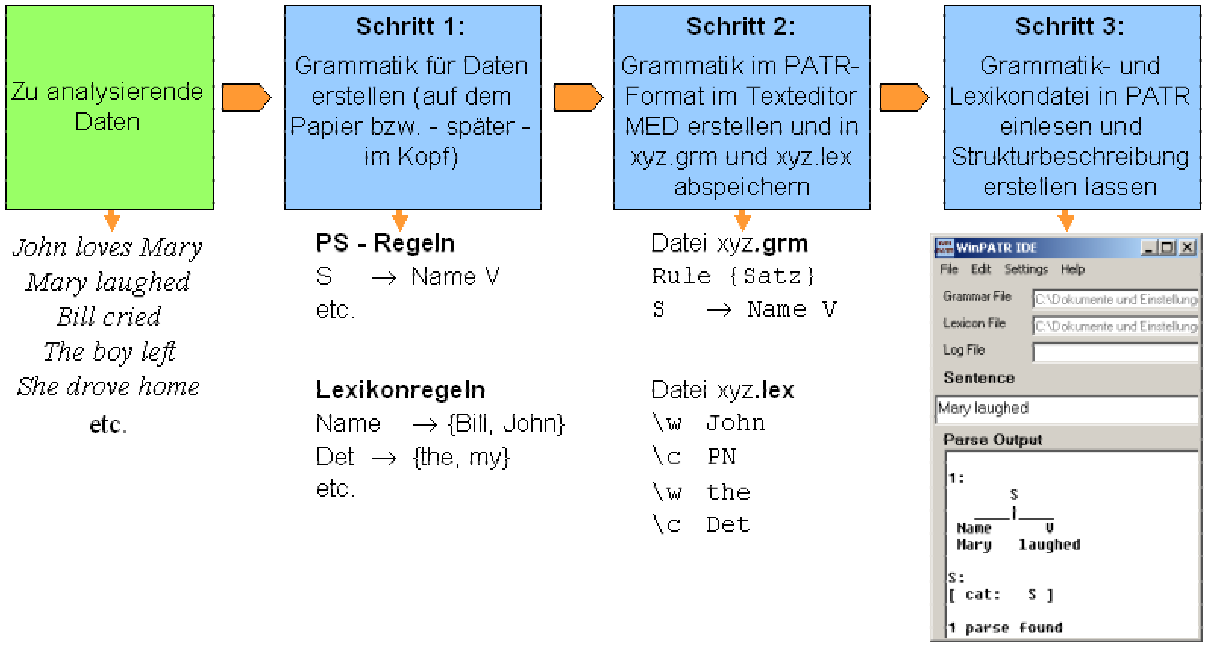


Abb. 3: Arbeitsablauf in PATR

Grammatik 0

Datensatz A:

Mary laughed	*Susan cried*
John jumped	*Tom ran*

1.1 Schreiben der Grammatik

Schritt 1. Schreiben Sie (auf Papier) eine Phrasenstruktur-Grammatik (PSG) für diese Daten. Sie besteht nur aus einer Regel: **S → Name V.** (Dies ist eine schlechte Grammatik. Warum? Kommt gleich).

Schritt 2. Umsetzung in PC-PATR-Form: Starten Sie das Programm **WinPatr**, am einfachsten über die Verknüpfung WIN PATR auf dem Desktop. Starten Sie von dort über den Menüpunkt **Editor > Start Editor** den Programm-Editor MED.

Schritt 3. Schreiben Sie die Grammatik im PATR-Formalismus und speichern Sie die Datei unter dem Namen psg0.grm in Ihrem Homeverzeichnis auf Z: (Menüpunkt **Datei > Speichern als**). Bestätigen Sie die Frage, ob der Dateiname übernommen werden soll, mit 'ja'.

Schritt 4. Legen Sie in MED eine neue Datei an. Erstellen Sie darin für Datensatz A das Lexikon in der Form wie oben angegeben und speichern Sie diese Datei unter dem Namen psg0.lex

1.2 Austesten der Grammatik

Nachdem Grammatik und Lexikon erstellt worden sind, testen wir, ob die Grammatik die Daten, von denen wir ausgegangen sind, korrekt analysieren kann.

Schritt 5. Wechseln Sie zum Programm **WinPatr** zurück und laden Sie die Grammatik psg0.grm über **File > Load Grammar**.

Schritt 6. Laden Sie über **File > Load Lexicon** die Lexikondatei psg0.lex.

Schritt 7. Geben Sie einen Satz aus den Ausgangsdaten in das Textfeld **Sentence** ein, z.B. *Mary laughed*. Sie müssen dafür die gleiche Orthographie verwenden wie im Lexikon und dürfen keine Interpunktionszeichen setzen.

Schritt 8. Lassen Sie den Satz analysieren, indem Sie auf den Schalter **Parse** klicken.

Wenn alles richtig ist, erscheint in dem Textfeld **Parse Output** ein Strukturbaum wie in **Abb. 3**.

Grammatik 1

Die Daten für **Grammatik 0** ließen sich mit einer einzigen Regel erfassen:

```
Rule   {einfacher Satz}
       S -> Name V.
```

D.h. dass wir die Grammatik exklusiv mit Bezug auf die in den Daten vorkommenden *lexikalischen Kategorien* (Wortarten) formuliert haben: *John, Mary, Susan, Tom* sind Eigennamen, *laughed, cried, jumped* etc. sind Verben.

Das ist allerdings unbefriedigend und wenig generalisierend. Was wir eigentlich ausdrücken müssen, sind die Feststellungen, dass ein Satz aus einer Nominalphrase (NP) und einer Verbalphrase (VP) besteht, und dass eine NP ein NAME sein kann und eine VP ein V sein kann. Wir müssen unsere Grammatik daher erweitern, indem wir folgende Regeln hinzufügen:

S → NP VP.

NP → Name.

VP → V.

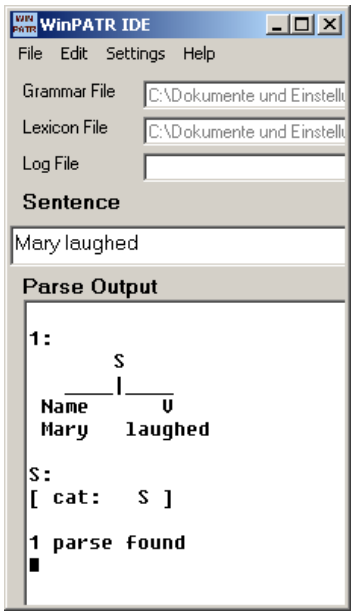
Schritt 9. Öffnen Sie die Grammatik psg0.grm zur Bearbeitung im Editor. Unter der Annahme, dass Sie diese Grammatik gerade mit **WinPatr** verwendet haben und selbige also geladen ist, geschieht dies am einfachsten, indem Sie im Menü **Edit** die Option **Edit Grammar File** wählen. Da sie wahrscheinlich ohnehin im Editor schon geöffnet ist, genügt es auch, einfach von **WinPatr** zu **MED** zu wechseln. Trifft dies alles nicht zu, müssen Sie den Editor starten und die Grammatikdatei öffnen.

Schritt 10. Speichern Sie die Grammatik unter dem neuen Namen psg1.grm ab.

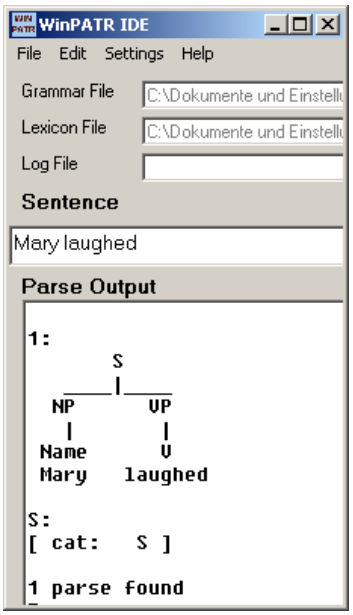
Schritt 11. Ersetzen Sie jetzt die einzige Regeln für den Satz durch die neuen Regeln für S, NP und VP im PATR-Format und sichern Sie sie.

Das Lexikon muss nicht geändert werden.

Grammatik 0 und Grammatik 1 im Vergleich



Parseergebnis für *Mary laughed* in **Grammatik 0**



Parseergebnis für *Mary laughed* in **Grammatik 1**

Wenn wir Grammatik 0 und Grammatik 1 vergleichen, stellen wir folgendes fest:

Die veränderte Fassung der Grammatik beschreibt die gleichen Daten wie die Grammatik 0. **Grammatik 0** und **Grammatik 1** sind diesbezüglich gleichwertig oder äquivalent. Grammatiken, die die gleichen Daten beschreiben, nennt man **schwach äquivalent**.

Grammatik 1 beschreibt zwar die gleichen Daten wie **Grammatik 0**, sie weist diesen aber eine andere, aussagekräftigere Strukturbeschreibung zu. Die beiden Grammatiken sind also in dieser Hinsicht nicht mehr äquivalent.

Grammatiken, die den gleichen Daten auch die gleichen Strukturbeschreibungen zuweisen, nennt man **stark äquivalent**.

Für unsere beiden Grammatiken trifft dies – wie gesehen – nicht zu.

Modifikation von Grammatik 1

Die Grammatikentwicklung soll im weiteren Verlauf als Modellkonstruktion betrieben werden, d.h. wir formulieren auf der Grundlage einer gegebenen Datenmenge eine Grammatik, die zu allererst die gegebenen Daten beschreiben muss und testen Sie dann an zusätzlichen Daten. Wenn diese zusätzlichen Daten nicht zufriedenstellend analysiert und beschrieben werden können, muss die Grammatik modifiziert werden. Auf diese Weise erhalten wir eine zunehmende Verbesserung und Verfeinerung unserer Grammatik.

Datensatz B

Mary saw John	*Susan slapped Mary*
John admired Susan	*Tom avoided John*

Schritt 12. Modifizieren Sie die Grammatik (zunächst in der üblichen Form auf Papier) so, dass auch diese Daten erfasst werden. Führen Sie zunächst keine neuen Kategorien ein.

Schritt 13. Setzen Sie Ihre Grammatik in die PATR-Form um. Sichern Sie das Ergebnis in psg1.grm.

Schritt 14. Ergänzen Sie das Lexikon psg1.lex mit den notwendigen Einträgen.

Schritt 15. Testen Sie Ihre Grammatik mit **WinPatr**.

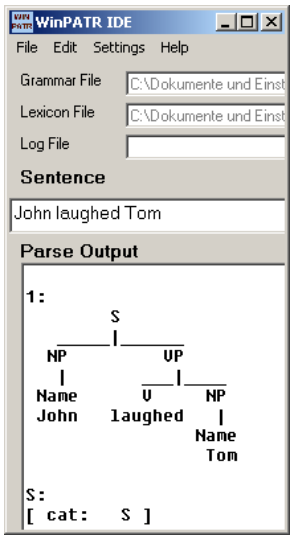
Wie verhält sich die Grammatik in der jetzigen Form gegenüber den folgenden ungrammatischen Ausdrücken?

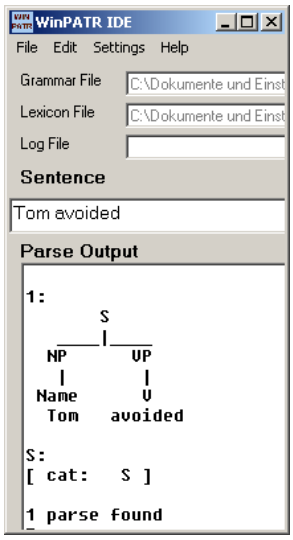
**John laughed Tom* **John admired*

**Susan cried John* **Tom avoided*

**Mary jumped Susan* **Mary saw*

Das Ergebnis ist, dass beispielsweise sowohl **John laughed Tom* als auch **Tom avoided* akzeptiert und analysiert werden. Unsere Grammatik ist also in dieser Hinsicht fehlerhaft:





Was läuft verkehrt? Nun, das Verb *avoided* darf nicht **ohne** Objekts-NP verwendet werden, das Verb *laughed* hingegen kann nicht **mit** Objekts-NP verwendet werden.

Wir müssen also für die Kategorie Verb wenigstens zwei Unterklassen, sogenannt **Subkategorien** ansetzen und diese in unserer Grammatik berücksichtigen. Einerseits haben wir Verben wie *laugh*, die kein Objekt zulassen, die sog. *intransitiven Verben*. Andererseits haben wir Verben wie *avoid*, die nicht **ohne** Objekts-NP stehen können, das sind die *transitiven Verben*. Das Verfahren der Unterteilung von Kategorien in Unterkategorien nennt man **Subkategorisierung**.

Für den Augenblick wollen wir auf eine Methode zurückgreifen, die in der Anfangszeit der generativen Grammatik sehr weit verbreitet war und darin bestand, für jede Subkategorie eine eigene Regel mit jeweils unterschiedlichen Kategorialsymbolen zu postulieren. Wir ordnen also den intransitiven Verben das Symbol **Vi** zu, den transitiven Verben das Symbol **Vt**.

Schritt 16. Ändern Sie die Regeln für die VP so, dass zwischen *intransitiven* (Vi) und *transitiven Verben* (Vt) unterschieden wird.

Schritt 17. Machen Sie die notwendigen Änderungen im Lexikon.

Schritt 18. Überprüfen Sie ob die modifizierte Grammatik die Probleme löst, d.h. Ausdrücke wie **John avoided* und **Mary cried Susan* ausschließt, hingegen aber *John avoided Tom* und *Mary cried* zulässt.

Grammatik 2

Modifizieren Sie Grammatik 1 so, dass auch die folgenden Daten korrekt beschrieben werden:

Datensatz C

the boy ran
the boy admired the girl
**the boy admired*
the student put the book on the table
**the student put the book*
**the student put on the table*
**the student put*

Mary gave John a present
**Mary gave John*
?Mary gave a present
**Mary gave*
the professor lived in a village
**the professor lived*

Hinweise:

1. Nehmen Sie als Grundlage die Grammatik [psg1.grm, psg1.lex] und speichern Sie diese unter den neuen Namen psg2.grm, psg2.lex.
2. Überlegen Sie genau, was an diesen Daten neu ist.
3. Verwenden Sie neben den schon eingeführten Kategorialsymbolen die folgenden Symbole: **Det** für Artikel *the, a*, **N** für Nomen, **P** für Präposition. Überlegen Sie, welche Subklassen von Verben vertreten sein müssen.
4. Beachten Sie, dass eine Phrase wie *on the table* eine syntaktische Einheit ist, die eine andere – *the table* – enthält.

Größere Datenmengen austesten

Wie wir gesehen haben, lautet ein wichtiges Grundprinzip von PATR wie folgt:

Eine korrekte Grammatik muss alle grammatischen Sätze erfassen und alle ungrammatischen ausschließen.

Wenn Sie also eine Grammatik erstellt haben, muss diese an **allen** Ausgangsdaten überprüft werden. Bei großen Datenmengen ist es aber eine lästige Angelegenheit, die zu analysierenden Daten bzw. Sätze immer wieder neu einzugeben.

Das Programm PATR bietet daher eine Möglichkeit, Sätze in einem Stapel zu verarbeiten. Dazu muss im Editor eine Textdatei angelegt werden, in der alle zu analysierenden Sätze enthalten sind. Diese Datei muss die Dateikennung **.sen** aufweisen.

Schritt 19. Erstellen Sie im Editor MED eine Satz-Datei mit den Daten aus Datensatz C. Geben Sie jeden Satz in eine neue Zeile und speichern Sie die Datei unter dem Namen **psg2.sen** ab. **Achtung**: In der Satzdatei dürfen weder Markierungen für ungrammatische ('*') bzw fragwürdige Sätze ('?') noch Interpunktionszeichen vorkommen!

Schritt 20. Laden Sie in PATR die Grammatik [psg2.grm, psg2.lex] (falls diese nicht ohnehin geöffnet ist)

Schritt 21. Klicken Sie in PATR im Menü *File* auf den Befehl *Parse File*. Dieser bietet die Möglichkeit, eine Liste von Sätzen, die in einer separaten Satzdatei enthalten sind, mit einem Befehl zu analysieren und das Analyseresultat in einer Ergebnisdatei abzuspeichern. Wählt man den Menüpunkt *Parse File*, so öffnet sich das Fenster zum Öffnen von Dateien, wobei allerdings nur Dateien mit der Erweiterung .sen gewählt werden können. Wählen Sie die Datei psg2.sen aus.

Schritt 22. Hat man eine Satzdatei ausgewählt, öffnet sich sofort erneut das Dialogfenster zum Öffnen von Dateien. Diesmal geht es darum, den Namen der Datei zu bestimmen, in die die Analyseergebnisse ausgegeben werden sollen. Es handelt sich ebenfalls um eine reine Textdatei. Die Standard-Dateityp-Erweiterung ist **.par**. Geben Sie als Namen **psg2.par** an. Es handelt sich hierbei um eine sog. Stapelverarbeitung: die Sätze in der Satzdatei werden nacheinander analysiert, und die Analyseergebnisse – falls erfolgreich – werden nacheinander in die Ausgabedatei (hier also psg2.par) geschrieben. Es erfolgt keine Ausgabe auf dem Bildschirm!

Schritt 23. Kehren Sie zum Editor MED zurück. Öffnen Sie die soeben von PATR erzeugte Analysedatei psg2.par und sehen Sie sich das Ergebnis an. Sie werden einen weiteren Vorteil dieser Vorgehensweise erkennen: im Gegensatz zur interaktiven Analyse wird bei der Stapelverarbeitung nämlich auch protokolliert, warum eine Analyse fehlschlägt.

Musterlösung zu Grammatik 2

Nachstehend finden Sie einen Lösungsvorschlag für Grammatik 2. Diesen sollten Sie nicht einfach abschreiben, sondern ihn

- während der Arbeit nur konsultieren, wenn Sie Probleme haben
- mit Ihrer eigenen, abgeschlossenen Grammatik 2 vergleichen und versuchen, die darin vertretenen Kategorien bzw. Strukturen nachzuvollziehen.

PS-Regeln (Datei psg2.grm):

```
Rule {Einfacher Satz}
S -> NP VP.

Rule {NP nur mit Eigenname}
NP -> Name.

Rule {NP mit Det und N}
NP -> Det N.

Rule {PP}
PP -> P NP.

Rule {VP intransitiv}
VP -> Vi.

Rule {VP transitiv}
VP -> Vt NP.

Rule {VP ditransitiv }
VP -> Vt2 NP NP.

Rule {VP mit NP und PP}
VP -> Vtp NP PP.

Rule {VP mit PP}
VP -> Vp PP.
```

Lexikonregeln (Datei psg2.lex):

Eigennamen

```
\w      Mary
\c      Name
\w      John
\c      Name
\w      Susan
\c      Name
\w      Tom
\c      Name
```

Nomina

```
\w      boy
\c      N
\w      girl
\c      N
\w      student
\c      N
\w      professor
\c      N
\w      book
\c      N
\w      table
\c      N
\w      village
\c      N
\w      present
\c      N
\w      ball
\c      N
```

Intransitive Verben

```
\w      laughed
\c      Vi
\w      jumped
\c      Vi
\w      cried
\c      Vi
\w      ran
\c      Vi
```

Transitive Verben

```
\w      saw
\c      Vt
\w      slapped
\c      Vt
\w      avoided
\c      Vt
\w      admired
\c      Vt
\w      kicked
\c      Vt
```

Ditransitive Verben

```
\w      gave
\c      Vt2
```

Verben mit P-'Objekt'

```
\w      lived
\c      Vp
```

Verben mit NP- und P-'Objekt'

```
\w      put
\c      Vtp
```

Determinatoren

```
\w      the
\c      Det
\w      a
\c      Det
```

Präpositionen

```
\w      on
\c      P
\w      in
\c      P
```

Grammatik 3

Modifizieren Sie Grammatik 2 so, dass auch die folgenden Daten korrekt beschrieben werden:

Datensatz D

the book was on the shelf
the book was quite interesting
the book was a flop
John lived in a very pleasant village
John lived in a pleasant village
**John lived in a very village*
John lived in a village
the rather silly boy gave the girl a precious present
the tall boy in the corner was my brother
the tree stood in the garden behind the house
the student watched the girl with a telescope
the student watched the girl
**the student watched with a telescope*

Kommentar zu Grammatik 3

Grammatik 3 [psg3.grm, psg3.lex] ist eine Modifikation der Grammatik 2, mit der auch die Formen aus Datensatz D zu beschreiben sein sollen. Was muss passieren, damit dieses Unterfangen gelingt?

1. Einführung neuer lexikalischer Kategorien: Kopulaverben

Relevante Daten: The book **was** { on the shelf / quite interesting / a flop }

Das Verb *was* kann eine PP als Ergänzung haben und ist insoweit wie *lived* (Vp) aus Grammatik 2. Es kann, wie ein transitives Verb, aber auch eine NP als Ergänzung haben, ist also ähnlich wie *watched* (Vt). Schließlich verbindet es sich auch mit einer Adjektivphrase (AP). Das Verb *was* ist aber weder ein Vp noch eine Vt. Was diese Fälle gemeinsam haben ist, dass die Ergänzungen nähere Bestimmungen (Modifikationen) zum Subjekt darstellen. In der traditionellen Grammatik spricht man von **prädikativen Ergänzungen** (sehen Sie im Text *(Some) Linguistic Foundations* auch die Angaben zu *Subject-Complements*).

Bei nominalen Ergänzungen zeigt sich der Sonderstatus darin, dass der Numerus nicht frei wählbar ist sondern mit dem des Subjekts übereinstimmen muss: man kann nicht sagen **the book was flops*. (Im Deutschen verhält es sich ebenso.)

Bei AP-Ergänzungen muss im Französischen das Adjektiv mit dem Genus des Subjekts übereinstimmen: *La maison est grande*. Wir müssen für das Verb *was* (und alle anderen Formen von *be*) eine eigene Subkategorie des Verbs annehmen. In der traditionellen Grammatik werden Verben wie *be* als **Kopula** (engl. *copula*, auch *linking verbs*) bezeichnet. Entsprechend nennen wir diese Klasse, zu der auch z.B. *become* oder *turn* gehören, **Vc**. und nehmen die entsprechenden Einträge im Lexikon vor.

In der nachstehenden Regel, die drei distinkten Regeln entspricht, wurde von der Abkürzungskonvention 'Schweifklammer' Gebrauch gemacht.

$$VP \rightarrow Vc \left\{ \begin{matrix} NP \\ AP \\ PP \end{matrix} \right\}$$

In PATR sieht das dann wie folgt aus (Alternativen sind durch den *Slash* voneinander unterschieden):

```
Rule   {VP mit prädikativer Ergänzung}
       VP -> Vc {NP / AP / PP}
```

2. Einführung neuer syntaktischer Kategorien I: AP in der NP

Relevante Daten:
John lived in a **very pleasant** village
John lived in a **pleasant** village
*John lived in a **very** village
John lived in a village

Hier geht es um eine fakultative Erweiterung der Struktur der Nominalphrase durch Ausdrücke wie *very pleasant*. Der ungrammatische Ausdruck zeigt, dass *very* nicht alleine stehen kann, sondern das Vorkommen eines Wortes wie *pleasant*, auf das es sich bezieht, voraussetzt: ein klassischer Fall von Dependenz, in dem das Adverb *very* vom Adjektiv *pleasant* abhängt. Dies ist ein Indiz dafür, dass *very pleasant* eine syntaktische Einheit bildet, deren Kopf das Adjektiv (A) *pleasant* ist. Wir haben es also mit einer ADJEKTIV-PHRASE (AP) zu tun. Wir benötigen folgende Regeln:

```
Rule   {AP ohne Adverb}
       AP -> A.
Rule   {AP mit Adverb}
       AP -> Adv A.
```

Bedenken Sie, dass Sie jetzt natürlich auch eine neue NP-Regel benötigen; nämlich eine, in der die NP mit einer AP auftritt:

```
Rule   {NP mit AP}
       NP -> Det AP N.
```

Abkürzungskonventionen

In Grammatik 3 haben wir nun jeweils zwei Regeln für die NP und die AP, die sich um jeweils ein fakultatives Element unterscheiden (eine fakultative AP in der NP; ein fakultatives Adverb in der AP). Per Konvention können diese beiden Regeln jeweils in einer einzigen ausgedrückt werden, indem das fakultative Element in Klammern gesetzt wird:

```
Rule   {NP}
       NP -> Det (AP) N.
Rule   {AP}
       AP -> (Adv) A.
```

2. Einführung neuer syntaktischer Kategorien II: PP in der NP

Relevante Daten: **the tall boy in the corner** was my brother
the tree stood **in the garden behind the house**

Im ersten Satz ist die PP *in the garden* eine nähere Bestimmung (Modifikation) zu *boy* und somit Bestandteil der NP *the tall boy in the corner*. Wir brauchen für die Beschreibung dieses Satzes also (a) eine Regel für die PP und (b) eine weitere Regel für die NP.

Die Regel für die PP lautete:

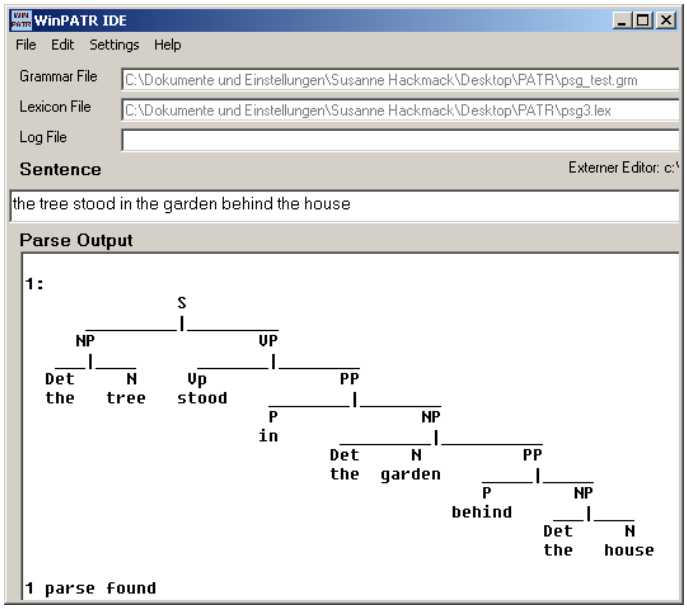
```
Rule   {PP}
       PP -> P NP.
```

Wir erweitern die oa. Regel für die NP um eine fakultative PP:

```
Rule   {NP mit Det, N und jeweils fakultativer AP und PP}
       NP -> Det (AP) N (PP).
```

Wenn wir das Lexikon entsprechend um die Präposition *behind* anreichern, erhalten wir für den ersten 'relevanten' Satz eine korrekte Beschreibung. Gleichzeit erzielen wir – sozusagen automatisch – auch die Analyse einer Lesart des zweiten Satzes, also *the tree stood in the garden behind the house.*

In dieser Analyse ist ausgedrückt. dass die PP *behind the house* Tochter der NP *the garden behind the house ist*. Hier also stehen die beiden PP *in the garden* und *behind the house* nicht auf einer Hierarchiestufe; stattdessen ist *behind the house* eine nähere Bestimmung zu *garden* ('der Garten, der sich hinter dem Haus befindet').



3. Einführung neuer syntaktischer Kategorien III: Fakultative PP in der VP

Relevante Daten: the student **watched the girl with the telescope**
the student **watched the girl**
*the student **watched with the telescope**

Diese Beispiele machen deutlich, dass die Ergänzung *with a telescope* keinesfalls obligatorisch ist, wohingegen das Objekt *the girl* nicht fehlen darf. Die *with*-Phrase bewirkt also keine Subkategorisierung der Verben. Tatsächlich hat sie den Status eines Adjunktes in der VP (sehen Sie zur Unterscheidung von Argumenten und Adjunkten den Text *(Some) Linguistic Foundations*). Man bezeichnet derartige PP auch als 'Instrumentalphrasen' (sehen Sie für eine Kurzeinführung in semantische Relationen den Text *(Some) Linguistic Foundations*).

Es wäre daher unangebracht, zu den bestehenden Regeln für Vi, Vp, Vt2, Vtp etc. noch jeweils weitere Regeln mit einer PP-Ergänzung hinzuzufügen. Das Verb *watched* ist in allen Beispielen ein einfaches transitives Verb Vt. Wir fügen, um die relevanten Daten zu beschreiben, die fakultative PP einfach der bestehenden Regel hinzu und verwenden als Notation die runden Klammern, die Optionalität bedeuten:

```
Rule   {VP transitiv mit fakultativer PP}
       VP -> Vt NP (PP).
```

Da wir weiter oben die NP-Regel um eine fakultative PP angereichert haben, haben wir somit auch erreicht, dass dem Satz *the student watched the girl with the telscope* nicht eine, sondern zwei Strukturbeschreibungen zugeordnet werden, und das ist recht und billig, haben wir es hier – wie sie sicher erkannt haben – wieder mit der Mutter aller strukturell mehrdeutigen Sätze zu tun.

Musterlösung zu Grammatik 3

Nachstehend finden Sie einen Lösungsvorschlag für Grammatik 3. Es sind nur diejenigen Einträge abgedruckt, die Grammatik 2 modifzieren bzw. erweitern.

Diesen Lösungsvorschlag sollten Sie nicht einfach abschreiben, sondern ihn

- während der Arbeit nur konsultieren, wenn Sie Probleme haben
- mit Ihrer eigenen, abgeschlossenen Grammatik 2 vergleichen und versuchen, die darin vertretenen Kategorien bzw. Strukturen nachzuvollziehen.

PS-Regeln:

```
Rule    {NP mit Det, N und jeweils fakultativer AP und PP}
        NP -> Det (AP) N (PP).
Rule    {AP}
        AP -> (Adv) A.
Rule    {VP mit Kopulaverb und NP bzw. AP bzw. PP Ergänzung}
        VP -> Vc {NP / AP / PP}.
Rule    {VP transitiv mit fakultativer PP}
        VP -> Vt NP (PP).
```

Lexikoneinträge:

Präpositionen

```
\w behind
\c P

\w with
\c P
```

Adverbien

```
\w quite
\c Adv

\w rather
\c Adv

\w very
\c Adv
```

Adjektive

```
\w silly
\c A

\w tall
\c A

\w precious
\c A

\w pleasant
\c A

\w interesting
\c A

\w awful
\c A
```

Kopulaverben

```
\w was
\c Vc
```