Programmierkurs Prolog

SS 1998

Thorsten Joachims

Universität Dortmund

LS VIII - Prof. K. Morik

nach N. Fuchs/T. Meyer

Uebersicht

- Hierarchie von Sprache
- Kontextfreie Sprachen
- DCGs in Prolog
- Unifikationsbasierte Grammatiken
- Strukturbeschreibungen
- Semantik

Sprachtypen

Sprachen können in eine Hierarchie nach ihrer Komplexität eingeordnet werden.

Rekursive Sprachen

$\uparrow$

Kontextsensitive Sprachen

$\uparrow$

Kontextfreie Sprachen

$\uparrow$

Reguläre Sprachen

Zu jeder Sprachfamilie gibt es einen Grammatiktyp.

Begriffe:

- Nichterminale Symbole
- Terminale Symbole
- Regeln

Parser

Ein Parser analysiert Sätze und gibt aus, ob diese wohlgeformt bzgl. einer gegebenen Grammatik sind.

Akzeptor: Antwortet nur ja/nein.

Komplexere Parser geben zusätzlich auch eine Strukturbeschreibung und/oder eine semantische Repräsentation des Satzes in einer semantischen Repräsentationssprache aus.

Kontextfreie Sprachen

Kontextfreie Sprachen werden durch Grammatiken beschrieben, die aus einer Menge von Regeln der Form

```
N --> V1, V2, ... , Vn
```

bestehen. N ist ein nichtterminales Zeichen und die V1 bis Vn sind entweder terminale oder nichtterminale Zeichen.

Mit kontextfreien Sprachen lassen sich die größten Teile von

- Programmiersprachen
- natürlichen Sprachen

beschreiben.

Prolog Regeln als Grammatik

Differenzlisten S\R:

```
s(S,R) :- np(S,R1), vp(R1,R).
np(P,R) :- det(P,P1), n(P1,R).
np(P,R) :- pn(P,R).
vp(P,R) :- tv(P,R1), np(R1,R).
pn(N,R) :- N=[mary|R].
pn(N,R) :- N=[tom|R].
tv(N,R) :- N=[loves|R].
det(N,R) :- N=[the|R].
n(N,R) :- N=[cat|R].
```

Aufruf:

```
?- s([tom,loves,mary],R).
```

Prolog als Parser

Prolog geht wie folgt vor:

- Top-Down: beginnend mit S werden die Grammatikregeln angewandt, bis terminale Symbole der Grammatik erreicht sind, die die Eingabe des Parsers bilden.
- Nichtdeterministische: wenn es mehrere Regeln mit demselben Kopf gibt, wird vielleicht zunächst eine angewandt, die nicht zum Ziel führt --> Backtracking.
- Depth-First: Tiefensuche ohne Tiefenbeschränkung
- Left-To-Right

==> Recursive Descent Parsing

Definite Clause Grammar

Phrasenstrukturgrammatik, die Kontextabhängigkeiten und beliebige Bedingungen verarbeiten kann.

Sie baut bei der syntaktischen Analyse im selben Schritt die syntaktische Struktur auf.

Der Formalismus ist genauso ausdrucksstark wie Prolog!

Regeln der Form:

```
NT --> Rumpf
```

Im Rumpf können beliebig viele Nichtterminale, Bedingungen und Terminale sein.

```
s --> np(nom,Num,Eig), vp(3,Num,Eig).
```

entspricht

```
s(X,Y):-np(nom,Num,Eig,X,X1),vp(3,Num,Eig,X1,Y).
```

X,Y,X1 sind Differenzlisten

DCGs in Prolog

Der Pfeil der Grammatikregel ist '-->'

Alle Terme entsprechen der normalen Prolog Syntax.

Terminale werden in eckigen Klammern [...] geschrieben.

Prolog Ziele in Grammatikregel werden in geschweifte Klammern {...} eingefaßt.

```
s --> np, vp.
np --> det, n.
np --> pn.
vp :- tv, np.
pn :- [mary].
pn :- [tom].
tv :- [loves].
det :- [the].
n :- [cat].
```

Aufruf:

```
phrase(s,[tom,loves,mary]).
```

Erzeugen der Strukturschreibung

Ein zusätzliches Argument:

```
s(s(NP,VP)) --> np(NP), vp(VP).
np(np(DET,N)) --> det(DET), n(N).
np(np(PN)) --> pn(PN).
vp(vp(TV,NP)) --> tv(TV), np(NP).
pn(pn(mary)) --> [mary].
pn(pn(tom)) --> [tom].
tv(tv(loves)) --> [loves].
det(det(the)) --> [the].
n(n(cat)) --> [cat].
```

Aufruf:

```
?- phrase(s,[tom,loves,mary]).

S = s(np(pn(tom)), vp(tv(loves),
np(pn(mary))))
```

Semantik

--> Tafel