Übungen 8: Dynamische Prädikate und Charts

Programmiertechniken in der Computerlinguistik II · Sommersemester 2005

Programmtexte finden sich auf Homepage: http://www.cl.unizh.ch/siclemat/lehre/ss05/pcl2/

1. Charts bauen

Gegeben sei folgender Grammatik-Ausschnitt:

NP → Det N	*NP → Adj N*	*N → old*	*Adj → old*
NP → Det Adj N		*N → men*	*Det → the*

Zeichne eine vollständige Chart mit passiven und aktiven Kanten für die Zeichenkette "the old men" als Graph und als Tabelle. Lass Einträge der Form LHS → . RHS weg.

2. Positionen: Numerisch oder listenförmig

Die Position, die ein Wort in einer Eingabekette einnimmt, kann sowohl durch Zahlen wie durch Differenzlisten ausgedrückt werden (vgl. Folie 7).

a) Definiere ein Prädikat assert_positions/1, das eine Liste von Wörtern entgegenimmt, und daraus die entsprechenden numerischen connects/3 assertiert.

```
?- assert_positions([the,dog,sleeps]).
yes
?- listing(connects/3).
connects(the, 1, 2).
connects(dog, 2, 3).
connects(sleeps, 3, 4).
yes
```

Ein Faktum wie connects(dog,2,3) wäre dann zu lesen als: Das Wort "dog" verbindet die Position 2 mit 3.

Hinweise: Manchmal braucht es ein geeignetes Hilfsprädikat für eine elegante Lösung.

b) Baue ein Prädikat position_to_list/3, das in den ersten beiden Argumenten zwei numerische Chart-Positionen entgegennimmt und als 3. Argument die entsprechende Zeichenkette als Liste zurückliefert.

```
?- position_to_list(2, 4, Kette).
Kette = [dog,sleeps];
no
```

3. Backtracking × Dynamische Prädikate = ?

Welche und wieviele Lösungen produziert die Anfrage `?- f(X).` mit dem linken bzw. rechten Programm?

```
:- dynamic g/1.
f(1) :- asserta(g(2)).
f(X) :- g(X).
```

```
:- dynamic g/1.
f(X) :- g(X).
f(1) :- asserta(g(2)).
```

Welchen Zusammenhang zum parse/2-Prädikat des Top-Down-Chart-Parser gibt es?

4. Top-Down-Parser mit Chart ohne Vollständigkeit

Lade die Datei "td_chart.txt" von der Homepage herunter. Sie enthält das Top-Down-Chart-Verfahren ohne Vollständigkeits-Check mit folgendem Grammatikausschnitt:

```
rule(s, [np,vp]).
rule(np, [det,n]).
rule(np, [det,n,pp]).
rule(vp, [v,np]).
rule(vp, [v,np,pp]).
rule(pp, [p,np]).
```

```
word(det, a).
word(p, with).
word(n, dog).
word(n, bone).
word(v, chased).
```

a) Stelle folgende Anfrage, nachdem du dir überlegt hast, wieviele 'y' du auf dem Bildschirm erwartest. Was bedeutet das Resultat auf dem Bildschirm?

```
?- clear_chart, parse(vp, [chased,a,dog,with,a,bone]-[]), write(y), fail.
```

5. Top-Down-Parser mit Chart mit Vollständigkeit

Lade die Datei "td_chart_c.txt" von der Homepage herunter. Sie enthält das Top-Down-Chart-Verfahren mit Vollständigkeits-Check mit dem gleichen Grammatikausschnitt.

a) Konsultiere die Datei und stelle dieselbe Anfrage wie in Aufgabe 4). Was bedeutet das Resultat?

```
?- clear_chart, parse(vp, [chased,a,dog,with,a,bone]-[]), write(y), fail.
```

b) Überlege dir ohne Computer, in welcher Reihenfolge die Charteinträge gemacht werden. Teste deine Hypothesen mit Hilfe der geschwätzigen Version des Parsers – mach diesmal manuelles Backtracking:

```
?- clear_chart, C=vp, verbose_parse(C, [chased,a,dog,with,a,bone]-[]).
```

Der geschwätzige Parser gibt 3 Arten von Information aus:

- Eintrag einer Kante in die Chart
  ADDED IN CHART: edge(np,[a,dog,with,a,bone],[with,a,bone])
- Erfolgreiches Nachschlagen einer Kante in der Chart
  FOUND IN CHART: edge(np,[a,dog,with,a,bone],[])
- Eintrag eines Vollständigkeitsvermerks für eine Kategorie ab einer Position in der Eingabekette
  COMPLETED CATEGORY 'p' STARTING ON [with,a,bone]

c) Was passiert, wenn du die Regeln für die VP vertauschst?

d) **Freiwillig**: Der Wechsel der Repräsentation für die Eingabekette vom Differenzlisten- auf das numerische Format erfordert nur 2 Änderungen. Die 1. Klausel von parse/2 muss ersetzt werden (linke Spalte) und vor der Parse-Anfrage müssen die Positionen assertiert werden. Bringe die Veränderung an und teste. Wieso geht das so einfach?

```
parse(C, From-To) :-
   connects(Word, From, To),
   word(C, Word).
```

```
?- L = [chased,a,dog,with,a,bone],
   assert_positions(L),
   parse(vp, 1-7).
```