Übungen 7: Left-Corner-Parsen

Programmiertechniken in der Computerlinguistik II · Sommersemester 2005

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

1. Aufrufdiagramm

Erstelle ein Aufrufdiagramm zum Left-Corner-Parser ohne Links, das die gegenseitigen Aufrufsverhältnisse der Prädikate ausdrückt.

2. Left-Corner-Parsen ohne und mit Links

Lade die Datei "lcp.txt" von der Homepage herunter. Implementiere alle syntaktischen und lexikalischen Regeln der folgenden Grammatik ausser den Tilgungsregeln:

S → *NP VP*	*N* → can | man | men
NP → *Det N*	*Det* → a | ε
NP → *NP Conj NP*	*Adv* → quickly | ε
VP → *V Adv*	*Conj* → and
VP → *Vmod Vinf*	*Vmod* → can
	V → sleeps | explodes
	Vinf → explode | sleep

a) Zeichne und schreibe dir anhand einer Darstellung, wie du sie auf Folie 8 findest, den Analysevorgang für den Satz "A man and a can can explode". Du kannst dein Ergebnis durch den Aufruf des gesprächigen Parsers

```
?- verbose_lcp(s, [a,man,and,a,can,can,explode]-[])
```

überprüfen.

b) Füge alle Tilgungsregeln und die Klausel von lcp/2 für Tilgungsregeln ohne den Aufruf von lc_link/2 ein (Code von Folie 14). Analysiere den Satz "A man sleeps" und den Satz mit dem unbekannten Wort "A xelofilant sleeps". Damit manuelles Backtracking möglich wird, musst du mindestens eine Variable in der Anfrage haben:

```
?- C = s, lcp(C, [a,man,sleeps]-[]).
```

Wie manifestiert sich das Tilgungsregelproblem? Was genau passiert?

c) Bestimme die lc_link/2-Relation für unsere Grammatik, implementiere die lc_link/2-Fakten inklusive reflexive Hülle und baue den Aufruf von lc_link/2 in die lcp/2-Klausel für Tilgungsregeln ein. Versuche zu formulieren, warum die obigen Anfragen ein besseres Verhalten zeigen.

3. Generieren

Überlege dir, ob der Left-Corner-Parser wie etwa der DCG-Standard-Parser generieren kann. Teste deine Hypothese mit der Anfrage:

```
?- lcp(s, S-[]).
```

Erkläre das Verhalten und überlege dir, ob man irgendwie nachhelfen kann.

4. Vom Akzeptor zum Parser

Unser Left-Corner-Syntaxanalyse-Programm ist ein Akzeptor, das beurteilen kann, ob eine Eingabekette bezüglich einer Grammatik syntaktisch wohlgeformt ist oder nicht. Es kann aber nicht die syntaktische Struktur des Analysebaums ausgeben, wie es für einen Parser gefordert ist.
Erweitere dein Programm aus Aufgabe 2 so, dass als Argument der Grammatiksymbole der entsprechende Ableitungsbaum aufgebaut wird.

```
?- lcp(s(Baum), [a, man, sleeps]-[]).
Baum = s(np(det(a),n(man)),vp(v(sleeps),adv(''))) ;
no
```

5. Konjunktive Adverbien (freiwillig)

Im akademischen Englisch werden satzinitial oft Ausdrücke wie "therefore, however, instead" etc. verwendet. Erwin E. formuliert deshalb die Regeln:

S → *AdvSI S*
AdvSI → *however* | *therefore* | *instead* | ε

Implementiere die Grammatikerweiterung und teste mit

```
?- C = s, lcp(C, [therefore,a,can,can,explode]-[]).
```

und mache manuelles Backtracking.

Was ist hier ein Problem?