Übungen 6: Morphologie und Buchstabenbäume

Programmiertechniken in der Computerlinguistik II · Sommersemester 2005

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

1. DCG mit Komposition und Flexion

Der DCG-Formalismus kann selbstverständlich sowohl für Komposition wie Flexion eingesetzt werden. Lade die Datei "dcg_morph.txt" von der Homepage herunter, die den in der Vorlesung vorgestellten Ansatz für Nominalflektion enthält.

a) Schreibe die notwendigen DCG-Regeln, damit für die Stämme "kind", "lied", "bild" und "brot" zweigliedrige Nominal-Komposita mit Fugenelement erkannt werden. Du kannst dich am Vorgehen aus der Vorlesung "Komposition und Differenzlisten" orientieren:

[kind]+[er]+[lied] vs. *[kind]+[]+[lied]

Passe die Stamminformation dementsprechend an!

b) Integriere die Komposition mit der Flexion so, dass das Programm am Schluss z.B. folgende Nominalphrase erkennt, ohne dass "kinderlied" als Stamm vordefiniert ist:

```
?- phrase(np(Kas,Gen,Num), [die, kinderlieder]).
Kas=nom, Gen=s, Num=pl;…
```

c) **Freiwillig**: Teste, ob dein Programm auch NPs generieren kann.

```
?- phrase(np(Kas,Gen,Num), NP).
…
```

Mach einen Trace, falls Probleme auftauchen. Modifiziere das Programm so, dass du mit einem *failure-driven-loop* am Prompt alle NPs ausgeben kannst, die dein DCG-Morphologie-Programm generieren kann.

```
?- phrase(np(Kas,Gen,Num), NP), write(NP), nl, fail.
```

2. Buchstabenbäume

Auf der Homepage findest du die Datei "trie_morph.txt", welche die Programme und das Beispiel für Buchstabenbäume enthält.

a) Füge den Stammeintrag für "brot" in den bestehenden Buchstabenbaum n_stamm_ltree/1 ein:

```
n_stamm_ltree(
[[b,[i,[l,[d, n_stamm(1,s,_,'BILD')]]]],
 [h,[a,[u,[s, n_stamm(1,s,sg,'HAUS')]]],
    [ä,[u,[s, n_stamm(1,s,pl,'HAUS')]]]]]
).
```

b) Erstelle eine weitere Klausel des Prädikats n_endung_ltree/2, welche für das Flexionsparadigma von "brot" benötigt wird:

```
n_endung_ltree(1,
      [n_endung(nom,sg),
       n_endung(dat,sg),
       n_endung(akk,sg),
       [e,n_endung(dat,sg),
          [r, n_endung(nom, pl),
              n_endung(gen,pl),
              n_endung(akk,pl),
              [n, n_endung(dat,pl)]],
          [s, n_endung(gen,sg)]]]).
```

Teste deine Erweiterungen aus!

c) Freiwillig: Überlege dir, wie man Buchstabenbäume (bzw. ASCII-Code-Bäume) automatisch aus einer tabellarischen Klauseldarstellung gewinnen könnte.

```
n_endung(1, nom, sg, "").
n_endung(1, gen, sg, "es").
n_endung(1, dat, sg, "").
n_endung(1, dat, sg, "e").
n_endung(1, akk, sg, "").
n_endung(1, nom, pl, "er").
n_endung(1, gen, pl, "er").
n_endung(1, dat, pl, "ern").
n_endung(1, akk, pl, "er").
```