Übungen 3: Tokenizer

Programmiertechniken in der Computerlinguistik II · Sommersemester 2005

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

1. Tokenizer modifizieren

a) Wieso verhält sich der Tokenizer auf folgende Anfrage so, wie er sich verhält?

```
?- read_atomics(Resultat).
|: Tränen fließen.
```

b) Modifiziere den Tokenizer so, dass er das erwartete Verhalten zeigt und mindestens obigen Input intuitiv behandelt.
Tipp: Den Zeichenkode beliebiger Buchstaben kannst du so ermitteln:

```
?- get_code(Code).
|: Ê
Code = …
```

2. Datei tokenisieren

a) Wie findest du die Lösung von Alfons N. für das Problem des zeilenweisen Tokenisierens von Dateien?

```
tokenize_file1(File, Lines) :-
   see(File),
   catch(
         findall(Line, ( repeat, read_atomics(Line) ), Lines),
         existence_error(_,_,_,_,past_end_of_stream),
         true
   ),
   seen.
```

b) Was hältst du von der Lösung, die sich Paula P. ausgedacht hat?

```
tokenize_file2(File, Lines) :-
   see(File),
   findall(
     Line,
     ( catch(
             ( repeat, read_atomics(Line) ),
             existence_error(_,_,_,_,past_end_of_stream),
             fail
       )
     ),
     Lines
   ),
   seen.
```

3. Abkürzungen erkennen

Definiere ein Prädikat recognize_abbrevs/2, das im Output von read_atomics/1 Abkürzungesbestandteile erkennen und als ein zusammengezogenes Token zurückgeben kann.

```
?- read_atomics(Atoms), recognize_abbrevs(Atoms, Tokens).
|: Z.T. wurde m.E. mit gesuchten Beispielen gearbeitet.
Atoms = [z,'.',t,'.',wurde,m,'.',e,'.',mit,gesuchten,beispielen,
gearbeitet,'.']
Tokens = ['z.t.',wurde,'m.e.',mit,gesuchten,beispielen,
gearbeitet,'.']
```

Hinweis: Versuche die Abkürzungen möglichst geschickt darzustellen, um eine einfache Verarbeitung zu bekommen! (Hier soll insbesondere Pattern Matching sowie rekursive Listenprogrammierung geübt werden.)

4. Tokenizer für deutsche Zeitungstexte anpassen

Freiwilliges Miniprojekt für Interessierte:
Auf der Homepage findest du eine Ausgabe der ComputerZeitung im Textformat (`cz.txt`). Überlege dir, wie eine sinnvoll tokenisierte Form aussehen sollte und passe den Tokenizer von Covington entsprechend an. Du kannst auch auf den im Skript erwähnten Nachfolge-Tokenizer von Covington zurückgreifen.