Institut Intelligente Systeme
Leiter: Prof. Dr. E. Lehmann
Dietmar Lippold

Übungsblatt 9

Besprechungstermin: 20.12.04 - 23.12.04

Aufgabe 1

Die folgenden Prozeduren[1] lösen zwar unterschiedliche Probleme, ihre Grundstruktur ist aber recht ähnlich:

```
(define (summe args f)
  (cond ((null? args) 0)
        (else (+ (f (car args))
                 (summe (cdr args) f)))))

(define (produkt args f)
  (cond ((null? args) 1)
        (else (* (f (car args))
                 (produkt (cdr args) f)))))

(define (fuer-alle args pred)
  (cond ((null? args) #t)
        (else (and (pred (car args))
                   (fuer-alle (cdr args) pred)))))

(define (mind-eins args pred)
  (cond ((null? args) #f)
        (else (or (pred (car args))
                  (mind-eins (cdr args) pred)))))
```

1. Markieren Sie im Code die Stellen, wo sich die Prozeduren unterscheiden.
2. Beschreiben Sie das Gemeinsame. Suchen Sie einen passenden Begriff (Namen) dafür.
3. Definieren Sie eine verallgemeinerte Prozedur für diesen gemeinsamen Anteil mit zusätzlichen Argumenten für die Unterschiede.
4. Definieren Sie die gegebenen Prozeduren neu als Spezialisierungen der verallgemeinerten Prozedur.

   Vorsicht: `and` ist in Scheme keine Prozedur sondern eine Sonderform (mit besonderem Auswertungsverfahren). Verwenden Sie `(lambda (p q) (and p q))`. Analog für `or`.
5. Definieren Sie `my-map` als Spezialisierung der verallgemeinerten Prozedur aus Aufgabenteil 3. Sie soll das gleiche wie die vordefinierte Prozedur `map` tun.

Aufgabe 2 (schriftlich)

Ein Parser für kontextfreie Grammatiken liefert einen Syntaxbaum in Listendarstellung zurück. Ein Baumknoten wird dabei als Liste des abgeleiteten Nichtterminalsymbols und der Konstituenten dargestellt.

[1] abrufbar unter `http://www.informatik.uni-stuttgart.de/ifi/is/Lehre/Vorlesung/info3.html`, Abschnitt Übungsbl¨atter

Beispiel: Grammatik $S \longrightarrow XY; X \longrightarrow aXb; X \longrightarrow ab; Y \longrightarrow cY; Y \longrightarrow c$ erzeugt für das Wort $aaabbbccc$ den Ableitungsbaum (`testbaum` in Listendarstellung)

```
(S (X a (X a (X a b) b) b)
   (Y c (Y c (Y c))))
```

1. Zeichnen Sie die Baumdarstellung der Liste (Liste als Knoten, Elemente als Unterbäume) und den Ableitungsbaum (Symbol als Knoten, Konstituenten als Unterbäume). Wie läßt sich die eine Darstellung aus der anderen erzeugen?

2. Definieren Sie eine Prozedur `(tiefe baum)`, die die größte Tiefe (Länge des längsten Ableitungswegs) des Baums zurückgibt, wobei ein einzelnes Symbol die Tiefe 1 hat. Sie können dabei die Prozedur `symbol?` verwenden. Überlegen Sie sich, welche Art von Argument der Prozedur jeweils übergeben wird und wählen Sie gegegebenfalls einen passenderen Namen für den formalen Parameter. Kommentieren Sie Ihre Prozedur ausführlich.
   `(tiefe testbaum) => 5`

3. Definieren Sie eine Prozedur `(wort baum)`, die aus dem Ableitungsbaum das Ausgangswort (Folge der Terminale) rekonstruiert.
   `(wort testbaum) => (a a a b b b c c c)`

4. Definieren Sie eine Prozedur `(nt-menge baum)`, die die Menge der im Baum vorkommenden Nichtterminale bestimmt. Sie können dazu die Prozeduren zur Verarbeitung von Mengen von Übungsblatt 7 verwenden.
   `(nt-menge testbaum) => (s x y)`

5. Definieren Sie eine Prozedur `(regeln baum)`, die die Liste der angewandten Regeln bestimmt. (Die Regel $S \longrightarrow XY$ soll z.B. als `(S X Y)` dargestellt werden)

   ```
   > (regeln testbaum)
   ((s x y) (x a x b) (x a x b) (x a b) (y c y) (y c y) (y c))
   ```

Aufgabe 3

In Aufgabe 3 des letzten Aufgabenblattes war eine Liste von Wörtern gegeben. Zu dieser Liste sollen folgende Wörter hinzugefügt werden, um weitere Komposita zu analysieren.

```
(define *wortliste2* '("auto" "bau" "berg" "blatt" "grund" "haus"
  "katze" "kirche" "platz" "ski" "stueck" "reparatur"))
```

1. Da die Wortlisten sehr groß werden können, soll ein effizientes Verfahren zur Vereinigung der beiden Listen implementiert werden. Das aus der Vorlesung bekannte `merge`-Verfahren vereinigt zwei sortierte Listen so, daß die Ergebnisliste sortiert ist. Implementieren Sie die Prozedur `(my-merge l1 l2)`.

2. Einzelne neue Wörter können in eine sortierte Liste durch direktes Einfügen hinzugefügt werden. Schreiben Sie die Funktion `(insert-sort wort liste)`, die ein Wort an der alphabetisch richtigen Stelle in eine Liste einfügt.

3. In Lexika für die maschinelle Sprachverarbeitung werden die Einträge meist weiter strukturiert. Die Einträge können wiederum Listen sein. Deshalb verwendet man Selektoren, die das Wort eines Eintrags bestimmen. Möchte man die Prozedur auch auf Listen mit Elementen anderen Typs (nicht nur Wörter) anwenden, sollte der Vergleichsoperator angegeben werden können. Erweitern Sie die Prozedur `insert-sort` zu `(insert-eintrag eintr lexikon selek vgl)`, die zusätzlich als Parameter einen Selektor und eine Vergleichsprozedur für den Wert des Selektors hat.

4. Geben Sie den Aufwand der von Ihnen definierten Prozeduren in Abhängigkeit von der Mächtigkeit der Listen an.