
Grundlagen von Programmiersprachen

<http://www.informatik.uni-freiburg.de/proglang/teaching/ws2003-2004/gvps>

Übungsblatt 14

Abgabe: 9.02.2004

Aufgabe 1 (CPS — Einfügen im Suchbaum).

Die Prozedur `insert-tree n b` liefert als Ergebnis einen Suchbaum zurück, der aus b durch Einfügen einer Zahl n entsteht, ohne den ursprünglichen Suchbaum b zu verändern.

Implementiere die Prozedur `insert-tree` in Scheme

- (a) in CPS bzw.
- (b) mit Hilfe von `call/cc`.

wobei mit Hilfe von Fortsetzung der Fall optimiert ist, falls das einzufügende Element im ursprünglichen Suchbaum schon vorhanden ist.

Aufgabe 2 (CPS — Subset-Sum-Problem).

In der Vorlesung wurde eine Implementierung zur Lösung von Subset-Sum-Problemen besprochen (`subsetsum1`), die sämtliche Lösungen eines gegebenen Problems aufzählt. Hierbei werden bei der Suche sämtliche Lösungen gespeichert, indem per Update-Operation nach und nach Lösungen in einer gemeinsamen Referenzzelle ablegen werden.

Gib eine einfachere Implementierung des Algorithmus an, die ohne Referenzzellen auskommt, die die `work`-Prozedur jedoch ebenso unverändert benutzt.

Aufgabe 3 (Koroutinen — Gleichheit von Binärbäume bzgl. Preorder-Traversal).

Betrachte für diese Aufgabe Binärbäume, die aus Blättern (zusätzlich annotiert um eine ganze Zahl) oder Baumknoten, die zwei Teilbäume enthält, bestehen. Auf diesen Binärbäumen sei ein zweistelliges Prädikat `samePreorder b1 b2` definiert, das genau dann wahr ist, falls beide Bäume den gleichen Preorder-Traversal besitzen.

Ein naiver Ansatz dieses Prädikat zu implementieren, ist es, zuerst separat den Preorder-Traversal beider Bäume in Form zweier Listen zu berechnen, und anschliessend beide Listen zu vergleichen. Diese Lösung ist jedoch ineffizient bezüglich ihres Zeit- und Speicherverhaltens, da beide Bäume jedesmal komplett durchlaufen und beide Preorder-Traversals zusätzlich zwischengespeichert werden müssen. Unterscheiden sich die Bäume etwa schon im ersten Blatt, ist das Ergebnis dann schon bekannt.

Eine cleverere Lösung benutzt zwei Koroutinen, die beide jeweils einen der Bäume in Preorder durchlaufen. Kommt die erste Koroutine an einem Blatt an, wird ihre Fortsetzung abgespeichert und die zweite Koroutine fortgesetzt. Kommt diese ebenfalls am nächsten Blatt an, wird ihre Fortsetzung ebenfalls abgespeichert und die Zahlen beider Blätter verglichen. Sind beide Zahlen gleich, wird der Baumdurchlauf fortgesetzt, andernfalls kann der Vergleich sofort abgebrochen werden.

- (a) Implementiere das Prädikat `samePreorder` mit zwei Koroutinen in Scheme in CPS.
- (b) Implementiere `samePreorder` mit Koroutinen nochmals entweder in C oder in Java.

Aufgabe 4 (CPS-Interpreter — try/raise).

Erweitere den CPS-Interpreter aus der Vorlesung um zwei weitere Ausdrücke `throw e` und `try e catch n -> e'` zur Behandlung von Ausnahmen ähnlich den entsprechenden Java-Konstrukten.

Die Auswertung von `throw e` wirft eine Ausnahme, die als Attribut den Wert von e trägt. Nach Auswertung von e wird der aktuelle Kontext verlassen und mit dem dynamisch aktuellsten, passenden Exception-Handler fortgefahren.

Der Ausdruck `try e catch n -> e'` installiert im aktuellen Kontext einen Exception-Handler e' , der Ausnahmen mit Attribut n behandelt, um dann e auszuwerten.