

Übungen zur Vorlesung  
„Grundlagen der Programm- und Systementwicklung“

**Aufgabe 7.1 Rekursion und Induktion**

Schreiben Sie eine ML-Funktion `swap`, die einen knotenmarkierten Binärbaum spiegelt, d.h. an der Wurzel den linken mit dem rechten Unterbaum vertauscht, nachdem die beiden Unterbäume ebenfalls gespiegelt wurden. Beweisen Sie, daß  $\text{swap}(\text{swap}(t))=t$  gilt.

**Aufgabe 7.2 Fehlermodellierung**

Ein Geldwechsellautomat soll modelliert werden. Die Aufgabe besteht darin, einen Geldebetrag als Summe von Münzen vorgegebenen Werts auszudrücken. Dabei kann davon ausgegangen werden, daß zu jedem Münzwert potentiell unendlich viele Münzen zur Verfügung stehen. Bei Eingabe der vorhandenen Münzwerte 10, 5, 2 und des zu wechselnden Betrags 36 beispielsweise ist eine mögliche Ausgabe die Sequenz  $\langle 10\ 10\ 10\ 2\ 2\ 2 \rangle$ . Der Betrag 3 läßt sich mit diesen Münzwerten nicht wechseln.

Schreiben Sie ein ML-Programm, das die Aufgabe löst. Überlegen Sie sich,

- (1) wie Sie den Fall abdecken, daß ein Geldbetrag nicht gewechselt werden kann, und
- (2) wie Sie vermeiden, daß von einem Münzwert zuviel herausgegeben wird. Hinweis: Verwenden Sie denselben Ansatz wie in (1).

**Aufgabe 7.3 (H) Curryfizierung**

Die polymorphe Funktion `map` und die Funktionen `s`, `*` und `+` auf `Nat` sind wie folgt definiert:

`map = (f:α→β, y:List α) List α:`

**if (y=⟨⟩) then ⟨⟩ else if y=(x^xs) then f(x)^(map(f,xs)) fi fi**

`s = (x:Nat) Nat: succ(x)`

`+` = (x:Nat, y:Nat) Nat: **if (x=0) then y else if (x=s(z)) then s(z+y) fi fi**

`*` = (x:Nat, y:Nat) Nat: **if (x=0) then 0 else if (x=s(z)) then y+(z\*y) fi fi**

- a) Berechnen Sie `map(s, ⟨0 s(0) s(s(0))⟩)` und `map( (x:Nat) Nat: x*x, ⟨0 s(0) s(s(0))⟩ )`.
- b) Welche Typen haben die vier Funktionen?
- c) Welche Typen haben die vier Funktionen, wenn man sie curryfiziert?

**Aufgabe 7.4 (H) Sequentielle Kompletierung**

Wir betrachten flach geordnete Sorten  $\alpha^\perp = \alpha \cup \{\perp\}$  mit kleinstem Element  $\perp$ .

Eine Funktion  $f: \alpha^\perp \rightarrow \beta$  heißt *sequentuell*, wenn

$f$  konstant ist oder

$f(\perp) = \perp$  und  $f(x)$  eine sequentielle Funktion ist, für jedes  $x \in \alpha$ , sofern  $\beta = \alpha_1^\perp \rightarrow \beta_1$ .

- a) Wie überträgt sich diese Definition auf Funktionen  $f: \alpha_1^\perp \times \alpha_2^\perp \times \dots \times \alpha_n^\perp \rightarrow \alpha_{n+1}^\perp$ ?
- b) Kompletieren Sie die durch  $(a \leq b) = (\neg a \vee b)$  auf `Bool` definierte Funktion zu einer nichtstrikten sequentiellen Funktion. Lassen sich auf `Bool` die Funktionen `<`, `=` und `∨` derart sequentiell kompletieren, daß  $(a \leq b) = ((a < b) \vee (a = b))$  auf `Bool`<sup>⊥</sup> gilt?