Mathematisches Institut
der Universität München
H. Schwichtenberg
L. Crosilla, M. Seisenberger

Übungen zum Ferienkurs über Nichtnumerisches Programmieren

Im Gebiet der sogenannten "künstlichen Intelligenz" ist es oft notwendig, Suchalgorithmen zu implementieren. Wir betrachten folgendes Problem.

> Im Urwald kommen drei Missionare zusammen mit drei Kannibalen an einen Fluß. Am Ufer liegt ein Boot, das höchstens zwei Personen faßt. Aus Sicherheitsgründen ist die Überfahrt so einzurichten, daß sich niemals an einem Ufer Missionare zusammen mit einer Überzahl von Kannibalen befinden. Man bestimme alle Lösungen des Problems.

Zur Lösung erklären wir, was *Zustände* sein sollen. Darunter verstehen wir Tripel aus zwei Zahlen und einem Symbol `left` oder `right`. Die erste (zweite) Zahl gibt die Anzahl der Missionare (Kannibalen) am linken Ufer an, und das folgende Symbol legt fest, an welchem Ufer sich das Boot befindet.

Zum Beispiel bedeutet der Zustand `(3 3 left)`, daß sich 3 Missionare und 3 Kannibalen auf dem linken Ufer befinden und auch das Boot am linken Ufer liegt; `(2 0 right)` bedeutet, daß sich 2 Missionare und kein Kannibale auf dem linken Ufer befinden, und das Boot am rechten Ufer liegt. Zum Ablesen der Daten eines Zustands verwenden wir `(define mis car)`, `(define can cadr)`, `(define boat caddr)`. Wir definieren jetzt, wann ein Zustand zulässig heißt, sowie einige Operationen an Zuständen, z.B. "Transport eines Missionars". Falls eine solche Operation nicht ausführbar ist, wird `#f` zurückgegeben.

Aufgabe 29. Man vervollständige die folgenden Definitionen

```
(define (admis state)
  (and (or (<= (can state) (mis state)) (zero? (mis state)))
         ;sichere Situation am linken Ufer
       ??)) ;sichere Situation am rechten Ufer

(define (move1mis state)
  (if (eq? 'left (boat state)) ;Boot am linken Ufer
      (if (positive? (mis state)) ;ein Missionar links
          (list ;wird uebergesetzt
            (- (mis state) 1) (can state) 'right)
          #f)
      ;jetzt der Fall Boot am rechten Ufer
      ??))
```

Aufgabe 30. Man definiere entsprechend `(move2mis state)`, `(move1can state)`, `(move1can state)`, `(move2can state)`, `(movemiscan state)`,

Aufgabe 31. Man vervollständige die folgende Prozedur, die einen Suchalgorithmus liefert. Der Trick besteht darin, mit Erfolgsfortsetzungen zu arbeiten. Ferner wird wesentlich benutzt, daß die primitive Prozedur `or` ihre Argumente immer von links nach rechts auswertet und mit der Auswertung aufhört, sobald ein als wahr geltender Wert gefunden wurde.

```
(define (path state sc) ;sc = success continuation
  (if (admis state)
      (or (if ?? ;Loesung gefunden
              (begin
                (display (reverse (cons state sc))) (newline)
                   ;Loesung ausgeben
                (display "more solutions wanted? (y/n)")
                (newline)
                (if (eq? 'y (read)) #f #t))
              #f)
          (let ((next1 (move1mis state)))
            (if (or ?? ;next1 schon betrachtet
                    ??) ;next1 keine Loesung
                #f
                (path next1 (cons state sc))))
          (let ((next2 (move2mis state)))
            ??)
          ??)
      #f))
```

Was ist die Ausgabe bei Eingabe von `(path '(3 3 left) '())`?

Abgabetermin. Freitag, den 11. Oktober 2002, in der Übung oder (besser) per email an `seisenb@mathematik.uni-muenchen.de`