magische Quadrate mit Suchverfahren

Das Programm ist nicht „die“ Lösung für magische Quadrate. Da es für magische Quadrate auch direkte Verfahren gibt, sind Suchverfahren nicht die erste Wahl. Wir betrachten es also nur als Übung für funktionale und modulare Programmierung.
Als Datentyp verwenden wir hier eine einfache Liste, in die am Kopf die neuen Zahlen eingefügt werden. Soll die Liste jeweils für eine Fortentwicklung eines Teils des Quadrates stehen, müssen wir sie allerdings zur Überprüfung invertieren, damit die zuletzt eingegebenen Zahlen immer die nachfolgenden Positionen auffüllen: Das erste Element soll während des Füllens immer das erste bleiben und nicht durch die neuen ersetzt werden.
Die neuen Elemente werden nicht aus allen Elementen gebildet, sondern nur die im Quadrat noch nicht auftauchenden verwendet. Das kann auch beim oder nach dem Einbau getestet werden, die Funktion **moeglichkeiten** ist aber ein gutes Beispiel für eine Funktion, die einen Filter bildet.

```
(define
  (moeglichkeiten zahl quadrat)
  (cond
    ((zero? zahl) '())
    ((member zahl quadrat)
     (moeglichkeiten (sub1 zahl) quadrat))
    (else
     (cons zahl (moeglichkeiten (sub1 zahl) quadrat)))))
```

hole-zeile und hole-spalte

Die funktionen hole-zeile und hole-spalte müssen das Problem lösen, dass die Zeile oder Spalte ggf. nicht lang genug sind. Für fehlende Werte wird an diesen Positionen jeweils eine 0 zurück gegeben. Das erledigt die Hilfsfunktion **hole-eine**.

```
(define
  (hole-eine nummer quadrat)
  (if (< nummer (length quadrat)) (list-ref quadrat nummer) 0))
```

Intern werden bei hole-zeile und hole-spalte index: 1 .. kante und position: 1 .. kante mit den angegebenen Wertebereichen verwendet, auch wenn die Funktion list-ref: 0 .. laenge-1 verwendet. Der Unterschied zwischen hole-zeile und hole-spalte besteht allein im Zugriff durch die Indizes. Hier folgt nur die eine Variante für die Spalte, bei der eine Rekursion über die Positionen in der Spalte erfolgt:

```
(define
  (hole-spalte index quadrat kante)
  (let holen
    ((position kante)
     (spalte '()))
    (cond
      ((zero? position) spalte)
      (else
       (holen
        (sub1 position)
        (cons
```

```
          (hole-eine
           (+ (* (sub1 position) kante) index  -1)
           quadrat)
          spalte))))))
```

position und index getauscht

fehlerhaft?

Eine Zeile oder Spalte (zos) ist fehlerhaft,

- wenn sie vollständig ist und nicht die richtige Summe hat oder
- wenn sie unvollständig ist und eine zu große Summe hat.

Unvollständig ist sie, wenn die letzte Zahl eine 0 ist.

hinten eine 0 heißt „unvollständig"

```
(define
  (fehlerhaft? zos summe)
  (cond
    ((zero? (car (reverse zos))) ;
     (>= (apply + zos) summe))
    (else (not (= (apply + zos) summe)))))
```

zosfehler?

Diese Funktion ist ebenfalls interessant. Einmal erfolgt bei ihr gleichzeitig eine Rekursion über alle Zeilen und Spalten. Besonders interessant ist aber, wie deren Teilergebnisse für den jeweiligen Index typisch funktional miteinander und mit dem Ergebnis des rekursiven Aufrufs verknüpft werden[1].

```
(define
  (zosfehler? quadrat summe kante)
  (let durchgehen?
    ((index kante))
    (cond
      ((zero? index) #f)
      (else
       (or
        (fehlerhaft? (hole-zeile index quadrat kante) summe)
        (fehlerhaft? (hole-spalte index quadrat kante) summe)
        (durchgehen? (sub1 index)))))))
```

Hier wird zudem ausgenutzt, dass bei einer or – Verknüpfung nach dem ersten #t – Wert die Auswertung beendet werden kann, da nun der Gesamtwert bekannt ist[2]!

Verbesserung

Deutlich verbessern ließe sich die Schnelligkeit der Bearbeitung noch, wenn man berücksichtigt, dass – solange nicht schon vorher ein Fehler auftrat, was bei uns dazu führt, dass wir mit diesem Ansatz nicht weiter arbeiten – nur mit dem letzten Wert überhaupt ein Fehler auftreten kann, also auch nur der zu prüfen ist. Dies sei eine Anregung zum eigenen Experimentieren!

Der Rest ist eine „normale" Tiefensuche mit backtracking.

1 Wegen der or – Verknüpfung muss beim Rekursionsabbruch der Wert #f zurückgegeben werden.
2 In diesem Fall eben, dass ein Fehler auftrat.