Funktionale Analyse der Breitensuche im Graphen

Beschreibung der Datenstrukturen:

- graph → ((knoten-1 folgeknoten-1 ...) (knoten-2 folgeknoten-a ...) ...)
- wege-liste → ((k-1 k-2 ... k-n) (k-a k-b ...) ...),
  dabei stellt die Liste der Knoten in den Teillisten einen möglichen Weg in umgekehrter Reihenfolge dar.
  Die Wegeliste stellt die eigentliche Datenstruktur dar, sie ist eine Warteschlange von Wegen.
- die-nachfolger → Liste von Knoten, die vom aktuellen Knoten zu erreichen sind

Modularisierung:

- expandiere-knoten → expandiert den ersten Knoten des ersten Weges und fügt die entstehenden Wege hinten an (der bisherige erste Weg selbst wird entfernt).
- nachfolger → bestimmt die Liste der Nachfolgeknoten eines Knoten.
  Eine mögliche Optimierung besteht darin, dass nur solche Knoten eingebaut werden, die der Weg noch nicht enthält.
- breitensuche → die Funktion selbst arbeitet als Aufrufhülle, die eigentliche Rekursion wird von der lokalen Funktion bs übernommen.

Verzweigungen

- wege-liste leer → Problem ist nicht lösbar.
- nachfolger enthalten den Zielknoten → Weg mit dem Zielknoten als Lösung ausgeben.
- nachfolger leer → mit dem Rest der wege-liste weiter arbeiten.
- sonst – Fall: nächsten Weg expandieren.

Ein Programm dazu:

```
(define
  (breitensuche start ziel graph)
  (define
    (nachfolger knoten)
    (cdr (assoc knoten graph)))
  (define
    (bs wege-liste)
    (define
      (expandiere-knoten folge-knoten)
      (cond
        ((null? folge-knoten) '())
        (else
         (cons
          (cons (car folge-knoten) (car wege-liste))
          (expandiere-knoten (cdr folge-knoten))))))
    (cond
      ((null? wege-liste) #f)
      ((member ziel (nachfolger (caar wege-liste)))
       (reverse (cons ziel (car wege-liste))))
      (else
       (bs
        (append
         (cdr wege-liste)
         (expandiere-knoten (nachfolger (caar wege-liste))))))))
  (bs
   (cons (cons start '()) '())))
; Beispielaufruf
(breitensuche 'hund 'katze '((hund maus loch floh) (maus loch katze hund floh)
 (loch hund maus) (katze maus floh baum) (baum katze floh) (floh hund maus katze
baum) ))
```

Besser verständlich als mit named-let ???