

Prozessalgebra

Aufgabe 1-1 Koordination kletternder Affen (6 Punkte)

- a) Eine Verklemmung entsteht, wenn zwei Affen, die in verschiedene Richtungen klettern, sich gleichzeitig auf der Liane befinden.
- b) Bevorzugen der heraufkletternden Affen: Die Affen “oben” und “unten” folgen verschiedenen Strategien.

Oben: Losklettern (nach unten), wenn

- kein Affe gerade nach oben klettert
- unten kein Affe wartet

Unten: Losklettern (nach oben), wenn gerade kein Affe nach unten klettert.

Problem: Die Affen oben können verhungern, d.h. sie kommen nie zum Zuge.

- c) Herunterkletternde Affen werden bevorzugt: Analog.
- d) Mehrere Möglichkeiten.
- Prinzip Baustellenampel: Die Liane wird abwechselnd für nach oben bzw. nach unten kletternde Affen geöffnet.
 - Nur ein Affe darf die Liane betreten, und die Affen klettern in der Reihenfolge ihres Eintreffens.
 - Die Affen ziehen beim Eintreffen an der Liane eine Nummer und klettern in dieser Reihenfolge. Dabei können zwei oder mehrere Affen, die in der gleichen Richtung unterwegs sind, direkt hintereinander klettern.

Aufgabe 1-2 Filtern von Nachrichten in CSP (6 Punkte)

In beiden Teilaufgaben wird das Programm durch die parallele Komposition $[S \parallel F \parallel E]$ dargestellt.

- a) Sender: Schreibt solange Zeichen auf den Kanal o , bis das Leerzeichen erreicht wird. Beachte die Kommunikationsanweisung in dem Wächter. Beachte die Kommunikationsanweisung in dem Wächter. Durch das Versenden des Endezeichens wird sichergestellt, das Filter und Empfänger auch vom Ende der Übertragung informiert werden.

```
E: var n: char[];  
    k: nat;  
    k = 0;  
    *[ n[k] != '#' & o!n[k] --> k := k + 1]  
    o! '#'
```

Filter: Beim erhalten eines Leerzeichens wird kein Zeichen gesendet. Beachte die Abfrage $c != '#'$ im Wächter der Schleife; hierdurch wird die Terminierung garantiert.

```

F: var c: char;
  c := 'a' /* beliebiges Zeichen != '#' */
  *[c != '#' & o?c --> [ [c = ' ' --> nop
                        [] c != ' ' --> i!c
                      ]
  ]

```

Empfänger: Analog zum Sender

```

E: var n: char[];
  k: nat;
  c: char;
  k = 0; c := 'a';
  *[ c != '#' & i?n[k] --> k := k + 1; ]

```

- b) Sender und Empfänger bleiben unverändert; lediglich der Filter führt zwei Variablen mit, die auf das zuletzt gesendete bzw. das zuletzt empfangene Zeichen zeigen.

```

F: var n: char[];
  in, out: nat;
  c: char;
  in := 0; out := 0; c := 'a';
  *[ out < in & i!n[out] --> out := out + 1
    [] c != '#' & o?c --> [ c = ' ' --> nop
                          [] c != ' ' --> n[in] := c; c = c + 1
                        ]
  ]

```