

## 1 Thema: Semaphore

### 1.1 Binärer Semaphore (10)

In der Vorlesung wurden allgemeine Semaphore und binäre Semaphore eingeführt. Dabei ist ein binärer Semaphore identisch zu einem allgemeinen (d.h. zählenden) Semaphore, der eben nur von null bis eins zählen kann.

1. Zeigen Sie, daß der binäre Semaphore dieselbe Leistungsfähigkeit besitzt, wie der allgemeine Semaphore. Geben Sie hierzu (Pseudocode) Programmfragmente für die Funktionen Signal und Wait an welche binäre Semaphore(n) verwendet und somit die Funktionsweise der allgemeinen Semaphore implementiert.
2. Ihre Lösung muß in irgendeiner Weise eine Zählervariable bereitstellen. Zeigen Sie, daß Ihr implementiertes Verfahren beim Zugriff auf diese Variable die Anforderungen, die wir an wechselseitigen Ausschluss gestellt haben, erfüllt (siehe Skript).

### 1.2 Synchronisation (12)

In einem Elfenbeinturm leben fünf Philosophen. Der Tageszyklus eines jeden Philosophen besteht abwechselnd aus Essen und Denken. Die fünf Philosophen essen an einem runden Tisch, auf dem in der Mitte eine Schüssel voll Reis steht. Jeder Philosoph hat seinen festen Platz am Tisch, und zwischen zwei Plätze liegt genau ein Stäbchen. Das Problem der Philosophen besteht darin, dass der Reis nur genau mit zwei Stäbchen gegessen werden kann, wobei jeder Philosoph nur das direkt rechts bzw. links neben ihm liegende Stäbchen zum Essen benutzen darf. Das bedeutet, daß zwei benachbarte Philosophen nicht gleichzeitig essen können. Von einer korrekten Lösung des Philosophenproblems wird folgendes erwartet:

- Sicherung des wechselseitigen Ausschlusses zweier benachbarter Philosophen.
- Verklemmungsfreiheit: Keine unauflösbaren Wartezustände möglich.
- Aushungerungsfreiheit: Jeder Philosoph, der essen möchte, ißt irgendwann auch einmal (= Verhungerungsfreiheit)

1. Für eine Lösung des Philosophenproblems seien die folgenden fünf binären Semaphore vereinbart:

```
VAR staebchen: ARRAY [0..4] OF Semaphore(1);
```

Jeder Philosoph  $j$  ( $j \in \{0, \dots, 4\}$ ) führe den folgenden Algorithmus aus:

```
/* Philosoph j */  
LOOP  
  DENKEN;  
  P(staebchen[j]);  
  P(staebchen[(j+1) MOD 5]);  
  ESSEN;  
  V(staebchen[j]);  
  V(staebchen[(j + 1) MOD 5]);  
ENDLOOP;  
}
```

Zeigen Sie, daß der angegebene Algorithmus keine korrekte Lösung des obigen Synchronisationsproblems darstellt. (*2 P*)

2. Konstruieren Sie eine im obigen Sinne korrekte Lösung des Philosophenproblems unter Verwendung von (binären oder allgemeinen) Semaphoren. Wieviele Philosophen könnten theoretisch gleichzeitig gleichzeitig essen ? Schränken Sie bei ihrer Konstruktion die maximal mögliche Parallelität so wenig wie möglich ein. Begründen Sie, warum Ihr Algorithmus verklemmungs- und aushungerungsfrei ist. (10 P)

## 2 Thema: Nachrichten (12)

### 2.1 Synchronisation

Eine weitere Möglichkeit, die Synchronisation zwischen parallelen Abläufen zu erreichen, stellt der Nachrichtenaustausch dar. Zur Kommunikation zwischen Prozessen seien die folgenden Befehle vorhanden:

- `SEND(Empfänger, Nachricht)`

Der Sender schickt eine Nachricht an den Empfänger und kann sofort weiterarbeiten.

- `RECEIVE(Nachricht)`

Der Empfänger empfängt eine Nachricht. Ist keine an den Empfänger adressierte Nachricht vorhanden, so blockiert dieser bis zum Erhalt einer Nachricht.

Die verschickten Nachrichten setzen sich aus den Komponenten `nachricht.sender` und `nachricht.inhalt` (Datentyp jeweils String) zusammen. Die Philosophen haben nach langem Denken und (Semaphor sei Dank) verklemmungsfreiem Essen entschieden, daß sie die Semaphore verkaufen und einen magischen Tisch erfinden, der auf Zuruf (Nachrichtenaustausch) die Stäbchen an die Philosophen verteilt. D.h. Kommunikation erfolgt nur zwischen Tisch und Philosophen.

- Geben Sie einen Konstruktionsentwurf des Tisches und der Philosophen in einer programmiersprachlichen Notation an.
- Denken Sie daran das die Lösung den Anforderungen an wechselseitigen Ausschluß genügt.