AG BS
Jan Peleska, Aliki Tsiolakis

Blatt 2

Prozessverwaltung

Aufgabe 1: Monitore — 50%

Neben Semaphoren gibt es weitere Konzepte zum Erreichen von *mutual exclusion* ohne *busy waiting*. Eines davon sind **Monitore**, die darüberhinaus auch das Blockieren des Gesamtsystems verhindern können.
Ein Monitor ist ein Sammlung von Prozeduren, Variablen und Datenstrukturen, die in einem speziellen Modul zusammengefasst werden können. Wesentliche Charakteristika:

- Zu jedem Zeitpunkt kann nur *eine* Prozedur des Monitor-Moduls aktiv sein. (Die Idee dabei ist, dass dies durch den Compiler gewährleistet wird, der hierzu bei der Übersetzung z.B. eine binäre Semaphore einführt.)
- Spezielle **condition-Variablen**, die nur über Operationen `WAIT` und `SIGNAL` angesprochen werden können, werden zum Blockieren und Reaktivieren von Prozeduren verwendet.

  Genauer: Die Prozedur, die die Resource (z.B. den gemeinsamen Puffer) benötigt, führt `WAIT(c)` auf eine geeignete Condition-Variable c aus. Daraufhin blockiert sie, und ermöglicht damit die Aktivierung einer anderen Prozedur. Dieser Prozedur kann dann durch `SIGNAL(c)` die andere Prozedur reaktivieren. In unserem Fall betrachten wir dabei die Variante von Hansen, in der `SIGNAL(c)` nur als letzter Befehl einer Prozedur auftreten darf, um zu gewährleisten, dass die rufende Prozedur auch terminiert ist, wenn die schlafende reaktiviert wird.

  Zu beachten ist, dass die Condition-Variablen keine Zähler sind; es ist daher notwendig, dass das `SIGNAL(c)` immer *nach* dem `WAIT(c)` erfolgt.

1. Modellieren Sie einen Monitor für die Verwendung beim Producer/Consumer-Problem geeignet als System in FDR. Verwenden Sie dabei Semaphoren zum Prüfen, ob eine Prozedur des Monitors aktiviert ist. Dokumentieren Sie Ihre Lösung.
2. Prüfen Sie mit FDR nach, dass ihre Modellierung *mutual exclusion* und Deadlock-Freiheit garantiert.

Das folgende „Programm“ soll als Hinweis auf die Verwendung des Monitors dienen:

```
monitor ProducerConsumer
  condition full, empty;
  integer   count;

  procedure insert;
  begin
    if count = N then WAIT(full);
    insert_item;
    count := count + 1;
    if count = 1 then SIGNAL(empty);
  end;

  procedure remove;
  begin
    if count = 0 then WAIT(empty);
    remove_item;
    count := count - 1;
    if count = N - 1 then SIGNAL(full);
  end;

  count := 0;
end monitor;
```

```
procedure producer;
begin
  while true do
  begin
    produce_item;
    ProducerConsumer.insert;
  end;
end;

procedure consumer;
begin
  while true do
  begin
    ProducerConsumer.remove;
    consume_item;
  end;
end;
```

Aufgabe 2: „Cigaret Smoker's Problem“ — 50%

In einem Raum sitzen drei Kettenraucher und ein Verkäufer von Tabakwaren. Um eine Zigarette rauchen zu können, benötigt ein Raucher drei Zutaten, nämlich Tabak, Papier und Streichhölzer. Der erste Raucher hat eigenen Tabak, der zweite eigenes Papier und der dritte eigene Streichhölzer. Sei ferner angenommen, dass die Vorräte des Verkäufers unbeschränkt sind.
Die Aktivitäten beginnen damit, dass der Verkäufer zwei Zutaten auf den Tisch legt, und damit einem Raucher ermöglicht zu rauchen. Ist der damit fertig, weckt er den Verkäufer, der dann wieder zwei (beliebige) Zutaten auf den Tisch legt. Dadurch wird dann wieder ein Raucher aktiv.

1. Modellieren Sie dieses System in FDR und geben Sie Ihre dokumentierte CSP-Spezifikation an.
2. Welche Probleme treten dabei auf? Geben Sie ggf. eine dokumentierte Lösung an.
3. Formulieren Sie die erwünschten Eigenschaften des Systems, und weisen Sie mit FDR nach, dass Ihre Modellierung Sie erfüllt.

Aufgabe 3: Ringpuffer 50%

Verändern Sie die Modellierung des Ringpuffers aus der Vorlesung (siehe Beispiel `http://www.informatik.uni-bremen.de/agbs/lehre/ws0102/bs1/bs1-aufgaben/ringbuf_20011112.csp` auf folgende Weise:

- Der abstrakte Puffer `B(s)` hat jetzt eine Kapazität $max\_abs = max + 1$.
- Der konkrete Puffer wird um eine Variable (CSP-Prozess) `IS_FULL` mit Wertebereich `Bool` erweitert.
- Die modifizierten Versionen von `READ_CON` und `WRITE_CON` operieren so auf `IS_FULL`, dass hierdurch die volle Kapazität ($max + 1$) des Implementierungspuffers `CBUF` ausgenutzt werden kann.

1. Geben Sie die dokumentierte CSP-Spezifikation des veränderterten Ringpuffers in FDR-Syntax an.
2. Überprüfen Sie mit FDR die beiden Verifikationsbedingungen.

Abgabe: Bis Montag, 26. November 2001, in der Übung.

Die CSP-Spezifikationen sollen in jedem Fall in elektronischer Form (als Mail-Attachment an `tsio@informatik.uni-bremen.de`) abgegeben werden!
Die Abgabe des gesamten Übungsblattes kann bevorzugt auf Papier aber auch als Email (nur ASCII-Text, Postscript oder PDF) an genannte Email-Adresse erfolgen.

Bei jeglicher Form der Abgabe die Gruppennummer und die Namen aller Gruppenmitglieder nicht vergessen!