Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Harald Fecher · Ben Lukoschus · Immo Grabe

Verifikation nebenläufiger Programme

Sommersemester 2004 — **Serie 2** — 26. April 2004

Thema: Das TLV-Tool, Mutual Exclusion, Deadlock, Starvation

Ausgabetermin: 26. April 2004

Abgabe: 3. Mai 2004 (vor der Vorlesung im Schrein oder in der Vorlesung)

Material zur Vorlesung: http://www.informatik.uni-kiel.de/inf/deRoever/SS04/Modcheck/

Aufgabe 1 (6 Punkte) Geben Sie für folgende SPL-Implementierung von Petersons Mutual-Exclusion-Algorithmus für 2 Prozesse:

$$\textbf{local } y_1, y_2 \textbf{ : boolean where } y_1 = y_2 = 0$$
$$s \textbf{ : } \{1,2\} \textbf{ where } s = 1$$

$$P_1 :: \left[\begin{array}{l} l_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{l} l_1 : \textbf{Non-Critical} \\ l_2 : (y_1, s) := (1,1) \\ l_3 : \textbf{await } y_2 = 0 \vee s \neq 1 \\ l_4 : \textbf{Critical} \\ l_5 : y_1 := 0 \end{array}\right] \end{array}\right] \parallel P_2 :: \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{l} m_1 : \textbf{Non-Critical} \\ m_2 : (y_2, s) := (1,2) \\ m_3 : \textbf{await } y_1 = 0 \vee s \neq 2 \\ m_4 : \textbf{Critical} \\ m_5 : y_2 := 0 \end{array}\right] \end{array}\right]$$

ein passendes SMV-Programm für das Tool TLV an, einschließlich der Definitionen für *justice* und *compassion.* Benutzen Sie TLV, um *mutual exclusion, absence of deadlock* und *absence of starvation (accessibility)* nachzuweisen. TLV 4.14 kann auf den SUN-Rechnern der Informatik als `/home/softtech/tools/bin/tlv` aufgerufen werden.

Ein Beispiel für ein SMV-Programm findet sich in Kapitel 3 der TLV-Dokumentation und auf der Rückseite dieses Aufgabenzettels.

Schicken Sie die Textdatei mit dem SMV-Code und eine Textdatei mit den Anweisungen für TLV bis zum Abgabetermin per EMail an `bls+serie02@informatik.uni-kiel.de`

Aufgabe 2 (6 Punkte) Schreiben Sie das SPL-Programm aus Aufgabe 1 so um, daß der Test in beiden **await**-Anweisungen nicht mehr atomar ist, d. h., daß ein Interleaving des anderen Prozesses zwischen dem Testen von y_i und dem Testen von s möglich ist.

Begründen Sie, ob sich dadurch die Eigenschaften *mutual exclusion, absence of deadlock* und *absence of starvation* verändern. Begründen Sie außerdem, ob dies auch von der Reihenfolge der Tests (y_i vor s oder andersherum) im geänderten Programm abhängt.

Beispiel für ein SMV-Programm, wie es von TLV verwendet werden kann:

```
MODULE main
VAR
  y : boolean; -- the semaphore variable. It is assigned by both processes.
  proc[1] : process user(y); -- The two processes have interleaved execution.
  proc[2] : process user(y);

ASSIGN
  init(y) := 1;

JUSTICE
  !proc[1].loc=3, !proc[2].loc=3

COMPASSION
  (proc[1].loc = 2 & y > 0,proc[1].loc = 3),
  (proc[2].loc = 2 & y > 0,proc[2].loc = 3)


MODULE user(y)
VAR
  loc : {0,1,2,3,4};
ASSIGN
  init(loc) := 0;

next(loc) :=
  case
    loc in {0,3}    : loc+1;
    loc = 1         : {1,2};
    loc = 2 & y = 1 : 3;
    loc = 4         : 0;
    1 : loc;
  esac;

next(y) :=                    -- changes to the semaphore variable.
case
  loc = 2 & next(loc) = 3 : 0; -- turned off when moving from l_2 to l_3
  loc = 4 & next(loc) = 0 : 1; -- turned on when moving from l_4 to l_0
  1 : y;
esac;
```