
Ü0: Ampel

a) Erstellen Sie in einem Texteditor das NuSMV-Modell einer Ampel, die in jedem Schritt einen Zustandswechsel ausführt!
   1) Starten Sie NuSMV mit Ihrem Modul im interaktiven Modus und lassen Sie das Modell dazu generieren. Simulieren Sie die Ampel und lassen Sie sich den Pfad dieser Simulation anzeigen!
   2) Ergänzen Sie das Modell um eine Spezifikation zur Lebendigkeit und lassen Sie diese von NuSMV prüfen!

b) Nun soll Ihr Modell um eine weitere Ampel ergänzt werden, so dass beide Ampeln den Verkehr an einer Kreuzung regeln!
   1) Modellieren und simulieren Sie das System für den Fall, dass die Ampeln synchron schalten!
   2) Spezifizieren Sie in CTL die Sicherheitseigenschaft "mindestens eine Ampel muss immer rot sein" und lassen Sie diese von NuSMV prüfen!
   3) Kann es vorkommen, dass eine Ampel nie einen kompletten Zyklus durchläuft?
   4) Wie sieht es mit Sicherheit und Lebendigkeit für den Fall asynchroner Ampel-Prozesse aus?

Ü1: Wechselseitiger Ausschluss

Zwei nebenläufige zyklische Prozesse führen einige Zeit lokale Operationen (`idle`) aus, bis sie schließlich einen kritischen Abschnitt (`critical`) erreichen (`entering` ⇒ `critical` ⇒ `exiting`), den jeweils nur einer der beiden Prozesse gleichzeitig ausführen darf.

a) Prüfen Sie folgende Spezifikation auf Sicherheit und Lebendigkeit.

```
MODULE proc (semaphore)
VAR
        state : {idle, entering, critical, exiting};
INIT
        state = idle;
ASSIGN
        next(state) :=
                case
                        state = idle : entering;
                        state = entering & !semaphore : critical;
                        state = critical : exiting;
                        state = exiting : idle;
                        TRUE : state;
                esac;
        next(semaphore) :=
                case
                        next(state) = critical : TRUE;
                        state = exiting : FALSE;
                        TRUE : semaphore;
                esac;

FAIRNESS
        running;
```

```
MODULE main
VAR
        semaphore : boolean;
        p1 : process proc(semaphore);
        p2 : process proc(semaphore);
INIT
        semaphore = FALSE;
```

b) Ändern Sie das Modell von a) so, dass beide Prozesse völlig synchron ausgeführt werden! Wie sieht es nun mit Sicherheit und Lebendigkeit aus?

c) Modellieren und validieren Sie den Algorithmus zum wechselseitigen Ausschluss nach Peterson und Fischer durch asynchrone Prozesse.

$y_1, y_2, t_1, t_2 \in \{\texttt{true}, \texttt{false}, \bot\}$, initial $\bot$	
start_1 : (* Prozess 1 *) $t_1 := \textbf{if } y_2 = \texttt{false} \textbf{ then } \texttt{false} \textbf{ else } \texttt{true} \textbf{ fi}$ $y_1 := t_1$ **if** $y_2 \neq \bot$ **then** $t_1 := y_2$ **fi** $y_1 := t_1$ **wait while** $y_1 = y_2$ *Kritischer Abschnitt 1* $y_1 := \bot; t_1 := \bot$ **goto** start_1	start_2 : (* Prozess 2 *) $t_2 := \textbf{if } y_1 = \texttt{true} \textbf{ then } \texttt{false} \textbf{ else } \texttt{true} \textbf{ fi}$ $y_2 := t_2$ **if** $y_1 \neq \bot$ **then** $t_2 := \neg y_1$ **fi** $y_2 := t_2$ **wait while** $(\neg y_2) = y_1$ *Kritischer Abschnitt 2* $y_2 := \bot; t_2 := \bot$ **goto** start_2

Ü2: Erzeuger/Verbraucher-System

Als häufig vorzufindendes "Design Pattern" ist ein Erzeuger-Verbraucher-System zu modellieren. Zwischen den Prozessen wird ein zu Beginn leerer Puffer für maximal fünf "Produkte" vorausgesetzt. Die beiden Prozesse dürfen nicht gleichzeitig Schreib- bzw. Leseoperationen am Puffer ausführen (wechselseitiger Ausschluss).

a) Erstellen Sie ein NuSMV-Modell beider Prozesse! Modellieren Sie das Gesamtsystem unter dem Gesichtspunkt eines asynchronen Ablaufs (Prozesse!).

b) Prüfen Sie, ob Ihre Umsetzung sicher und lebendig ist, das heißt, ob der gegenseitige Ausschluss gewährleistet ist, ob jeder Prozess seinen kritischen Abschnitt jedes Mal ausführen kann sobald er das wünscht und ob jeder Prozess seinen kritischen Abschnitt beliebig oft ausführen darf!