

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 ($\text{entering} \Rightarrow \text{critical} \Rightarrow \text{exiting}$), den jeweils nur einer der beiden Prozesse gleichzeitig ausführen darf.

- a) Prüfen Sie folgende Spezifikation auf Sicherheit, Lebendigkeit und Fairness durch Simulation und mittels CTL-Formeln.

```
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;
            1 : state;
        esac;
    next(semaphore) :=
        case
            next(state) = critical : TRUE;
            state = exiting : FALSE;
            1 : 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 synchrone Prozesse.

$y_1, y_2, t_1, t_2 \in \{\text{true}, \text{false}, \perp\}$, initial \perp	
<pre> start₁ : (* Prozess 1 *) t₁ := if y₂ = false then false else true fi y₁ := t₁ if y₂ = \perp then t₁ := y₂ fi y₁ := t₁ loop while y₁ = y₂ <div style="border: 1px solid black; padding: 2px; text-align: center; margin: 5px 0;"><i>Kritischer Abschnitt 1</i></div> y₁ := \perp; t₁ := \perp goto start₁ </pre>	<pre> start₁ : (* Prozess 2 *) t₂ := if y₂ = true then false else true fi y₂ := t₂ if y₁ = \perp then t₂ := \negy₂ fi y₂ := t₂ loop while \negy₁ = y₂ <div style="border: 1px solid black; padding: 2px; text-align: center; margin: 5px 0;"><i>Kritischer Abschnitt 2</i></div> y₂ := \perp; t₂ := \perp goto start₁ </pre>

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!
- c) Erweitern Sie das Modell von a) so, dass Erzeuger und Verbraucher synchron ausgeführt werden und wiederholen Sie b)! Nutzen Sie dazu den Algorithmus von "Peterson & Fischer" für den gegenseitigen Ausschluss.
- d) Können Sie mit CTL nachweisen, dass vor jedem (ersten) "Verbrauchen" stets etwas "erzeugt" werden muss?