Formale Grundlagen der Informatik III: Lab 1 (SPIN)

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2014/2015
Deadline zur Abgabe: Montag, der 01.12.2014 - 23:59 Uhr

Im Rahmen dieses Labs müssen Sie Code in der Sprache **PROMELA** schreiben und zum Testen **SPIN** verwenden. Sollten Sie SPIN noch nicht installiert haben, beachten Sie bitte die Anleitung, die in Moodle unter dem Punkt **0 - Organisation** zu finden ist.
Zur Abgabe laden Sie bitte alle erstellten Codes gesammelt in einem .zip Archiv vor der angegebenen Deadline über das Moodle Portal der Veranstaltung hoch.

Über alle **3** Labs können sie eine maximale Menge von **100** Punkten erlangen. Durch das vorliegende Lab können Sie **30** Punkte erzielen.

Informationen zur Abgabe: Bitte benennen Sie die Dateien ihrer Abgabe sinnvoll, damit diese problemlos den jeweiligen Aufgaben zugeordnet werden können. Geben Sie bitte in den Kopf jeder .pml Datei (in Form eines Kommentars) und jeder Textabgabe die Mitglieder Ihrer Lab-Gruppe sowie die Gruppennummer an. Für Textabgaben verwenden Sie bitte ein reines Textformat (.txt).
Fassen Sie abschließend alle Daten ihrer Abgabe in einem .zip Archiv zusammen und bennenen Sie dieses nach folgender Konvention: lab1-l**xx**.zip, wobei **xx** durch Ihre jeweilige Gruppennummer zu ersetzen ist.

Hinweis: *Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weiter Informationen finden Sie unter: https://www.informatik.tu-darmstadt.de/de/sonstiges/plagiarismus/*

Aufgabe 1 Modellierung von Schaltkreisen (10 P.)

Im ersten Teil des aktuellen Labs befassen wir uns mit der Modellierung einiger einfacher Schaltkreise, die Sie aus den technischen Grundlagenveranstaltungen bereits kennen sollten.

Aufgabe 1.1 8 Bit Vergleicher (5/10 P.)

Erstellen Sie ein PROMELA Modell, welches einen 8 Bit Vergleicher abbildet. Im Modell werden zwei Bit Arrays a und b vorgegeben. Das Modell soll 1 ausgeben, wenn $(a_1 a_2 ... a_8)_2 \leq (b_1 b_2 ... b_8)_2$, wobei $(s)_2$ die Zahl angibt, die aus der Interpretation jedes binären Arrays resultiert. Andernfalls soll 0 ausgegeben werden.
Simulieren Sie das Modell und schreiben Sie entsprechende *inline Assertions* um das Programm zu verifizieren (es müssen **mindestens 3 sinnvolle Assertions** verwendet werden).

Aufgabe 1.2 8 Bit Addierer (5/10 P.)

Schreiben Sie nun einen 8 Bit Addierer. Dem Addierer werden zwei Bit-Arrays (siehe Aufgabe 1.1) gegeben. Die Ausgabe sollte sowohl das Resultat der Berechnung enthalten, als auch eine Markierung für das Carry-Bit.
Simulieren Sie das Modell und erstellen Sie entsprechende *inline Assertions* (es müssen auch hier **mindestens 3 sinnvolle Assertions** verwendet werden).

Aufgabe 2 Token Ring (10 P.)

Ein Token Ring besteht aus **m** unabhängigen Prozessen, die in einem Kreis angeordnet sind. Hierbei ist jeder Prozess mit seinem linken und rechten Nachbar verbunden. Die Prozesse des Token Rings nutzen ein Token (dargestellt durch eine boolsche Flag) um sich zu synchronisieren. Nach jedem Berechnungsschritt wird das Token an einen Nachbarn weitergereicht.
Hinweis: *Bei den folgenden Aufgabenteilen* ***b) - d)*** *müssen Sie verschiedene Korrektheitseigenschaften nachweisen. Formulieren Sie hierfür entweder passende Assertions oder LTL-Eigenschaften und nutzen Sie diese anschließend zur Verifikation mittels SPIN.*

a) Implementieren Sie einen Token Ring für m = 4, wobei das Token nicht-deterministisch an einen der beiden Nachbarn weitergereicht wird.
Simulieren Sie den Token Ring interaktiv um seine Funktion zu prüfen.

b) Nutzen Sie SPIN um sicherzustellen, dass jeder Prozess das Token unendlich oft erhält.

c) Stellen Sie sicher, dass in Ihrem Modell niemals zwei Prozesse zur gleichen Zeit Zugriff auf das Token haben können.

d) Betrachten wir nun den Fall, dass in dem Ring zwei Tokens existieren. Jeder Prozess kann nun zwei Tokens gleichzeitig halten. Zum Durchführen seiner Berechnungen benötigt ein Prozess weiterhin mindestens ein Token. Nach jedem Berechnungsschritt übergibt der Prozess alle vorhandenen Tokens nicht-deterministisch an seine Nachbarn. Erstellen Sie ein neues Modell, welches dieses Verhalten in PROMELA abbildet.
Betrachten Sie anschließend die folgenden Aufgabenstellungen:

   1. Verifizieren Sie, dass jeder Prozess zumindest ein Token unendlich oft erhält.
   2. Erhält jeder Prozess unendlich oft beide Token zur gleichen Zeit? Wenn ja, so weisen Sie dies auch durch Verifikation nach.

Aufgabe 3 Labyrinth (10 P.)

Gegeben sei das Labyrinth wie in Abbildung 1 zu sehen. Lassen Sie durch SPIN einen Pfad von Punkt **A** zu Punkt **B** identifizieren *(es sind nur horizontale und vertikale Bewegungen möglich!)*.

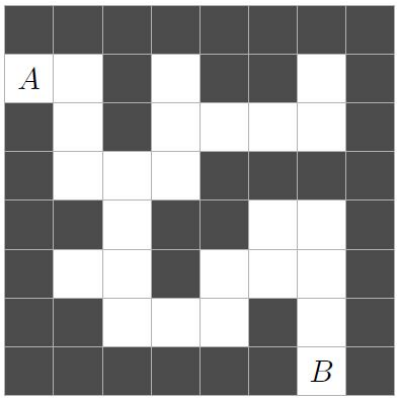

Abbildung 1: Labyrinth