Übungen zur Vorlesung
Betriebssysteme I
Studiengang Informatik
WS 08/09

AG Betriebssysteme FB3
Jan Peleska, Florian Lapschies

Ausgabe: 06.11.2008
Abgabe: 27.11.2008

Übungszettel 1

Hinweise

Die Abgabe erfolgt als Ausdruck am Ende der Vorlesung und als E-Mail an *florian@informatik.uni-bremen.de*. Die Dokumentation der Aufgabenlösung ist in LaTeX anzufertigen. **Auf jeden Fall** sollte die elektronische Abgabe alle mit UPPAAL erstellten Dateien umfassen. Der Betreff der E-Mail sollte folgendes Aussehen haben:
BS1 Abgabe x Gruppe y.

Bitte immer die Namen aller Gruppenmitglieder und die Gruppennummer angeben!

Aufgabe 1: Petersons Algorithm

In der Vorlesung wurde euch *Petersons Algorithmus* zum gegenseitigen Ausschluss von zwei auf einen kritischen Abschnitt zugreifenden Prozessen vorgestellt. Dieser Algorithmus ist in Listing 1 abgedruckt. Hierbei sind beide Prozesse über eine Eindeutige Prozess-ID aus $\{0, 1\}$ identifiziert. Beide Prozesse rufen vor dem Betreten des kritischen Abschnitts enter_region() und nach dem Verlassen leave_region() mit ihrer jeweiligen Prozess-ID auf.

Listing 1: Petersons Algorithmus

```
1  volatile int turn;
2  volatile bool interested[2] = {false, false};
3
4  /* pid ist aus {0,1} */
5  void enter_region(int pid) {
6          interested[pid]= true;
7          turn = pid;
8          while (turn == pid && interested[1-pid]);
9  }
10
11 void leave_region(int pid) {
12         interested[pid] = false;
13 }
```

a) Erläutert die Funktionsweise dieses Algorithmus.

b) Modelliert den Algorithmus mit dem UPPAAL tool.

c) Stellt die nötigen Anforderungen für die Korrektheit des Algorithmusses auf und prüft mit UPPAAL das Modell gegen die Anforderungen.

d) Ist der Gegenseitige Ausschluss weiterhin gewährleistet, wenn Zeile 6 und 7 in dem Algorithmus vertauscht werden? Erläutere den Grund hierfür. Modelliert diesen abgeänderten Algorithmus mit UPPAAL. Gebt gegebenenfalls eine Ausführungsreihenfolge an, die das Gegenteil beweist und fügt die zugehörige Trace-Datei der elektonischen Abgabe hinzu.

Aufgabe 2: Strict Alternation

In der Vorlesung wurde der *Strict Alternation* Algorithmus und dessen Modellierung mit UPPAAL vorgestellt. Die Spezifizierung eines zugehörigen Timed Automata aus der Vorlesung findet ihr in dem Archiv *strict_alternation.tar.gz* in Form der Dateien *strict_alternation.xml*.

a) Bei dem Strict Alternation Algorithmus kann einer der beiden Prozesse nur die Critical Section betreten, wenn die *turn* Variable zuvor durch den anderen Prozess auf seine Prozess-ID gesetzt wurde. Erläutert warum bei dieser Modellierung beide Prozesse stets vom dem Zustand *wait* auch schließlich zu dem Zustand *cs* kommen, wie in den beiden ersten beiden Verifikationsbedingungen spezifiziert.