Übungen zur Vorlesung
Betriebssysteme I
Studiengang Informatik
WS 2010/2011

AG Betriebssysteme FB3
Prof. Dr. Jan Peleska, Florian Lapschies

Abgabetermin: 20.12.2010

Übungszettel 3

Die Speisenden Philosophen

In Vorlesung wurde Euch das Konzept von Semaphoren und Mutexen, sowie die damit einhergehende Gefahr von Deadlocks vorgestellt. Ein gängiges illustratives Fallbeispiel für diese Gefahr ist das sogenannte *Philosophenproblem*:

Fünf Philosophen sitzen gemeinsam an einem runden Tisch. Vor jedem Philosophen liegt jeweils ein großer Teller voller Nudeln. Zwischen zwei nebeneinander sitzenden Philosophen liegt jeweils eine Gabel, so dass es also am Tisch genauso viele Gabeln wie Philosophen gibt. Jeder Philosoph *denkt* entweder oder er *ißt*. Zum Essen nimmt sich ein Philosoph nacheinander die beiden Gabeln zu seiner Seite. Nach dem Essen legt die Gabel wieder nacheinander auf dem Tisch und denkt daraufhin wieder.

Dieses Verhalten kann so natürlich zu einem Deadlock führen, wenn ein jeder Philosoph die Gabel zu seiner linken genommen hat und dann darauf wartet, dass die Gabel zu seiner rechten ebenfalls verfügbar wird.

In dem der Aufgabe beiliegenden C-Programm haben nebenläufig aggierende Threads die Rolle der Philosophen. Die Gabeln wurden realisiert durch Mutexe, wobei ein Mutex geschlossen ist, wenn die zugehörige Gabel gerade von einem Philosoph genutzt wird, andernfall ist er offen.

Aufgabe 1

Erklärt, wie in diesem Programm das Philosophenproblem gelöst wurde.

Aufgabe 2

Modelliert diese Lösung in UPPAAL und weist nach, dass es auch tatsächlich zu keinem Deadlock kommen kann.

Berücksichtigt dabei die folgenden Hinweise:
Definiert Euch eine Konstante `const int N = 5`, mit der Ihr die Anzahl der Philosophen festlegt. Erstellt dann ein Prozess-Template `Philosoph(int[0,N-1] id)` und modelliert dort das Verhalten eines Philosophen mit der Prozess-ID `id`. Des weiteren erstellt Ihr ein Template `Semaphore(int[0,2*N] id, int init)` und modelliert dort das Verhalten einer zählenden Semaphore mit der ID `id` und dem initialen Zählerwert `init`. Deklariert die Kanäle `chan down[2*N+1]` und `chan up[2*N+1]`, so dass die Philosophenprozesse über `down[i]!` bzw. `up[i]!` die Semaphore `i` ansprechen können. Natürlich akzeptiert eine Semaphore nur ein *down*, wenn der zugehörige Zähler noch nicht Null erreicht hat.

Hinweise

Die Abgabe erfolgt als Ausdruck am Ende der Vorlesung und zusätzlich elektronisch über das Subversion Repository. Die Dokumentation der Aufgabenlösung ist in LaTeX anzufertigen. Bitte vergesst nicht die Namen aller Gruppenmitglieder mitanzugeben.