Übung: Real-Time Model-Checking

Zielstellung der Übung

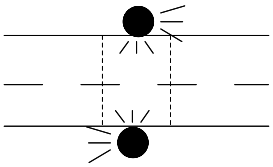
In der Übung sollen Sie sich mit dem Model-Checking-Tool UPPAAL vertraut machen. Zu diesem Zweck werden Sie zunächst ein Beispielmodell untersuchen

Im zweiten Teil der Aufgabe werden Sie eine Ampelschaltung für eine Fußgängerampel modellieren. Teile der Lösung sind dabei schon vorgegeben. Sie werden Anforderungen an das Modell in Form von Queries formulieren und diese prüfen, sowie das Modell im Simulator des Tools untersuchen.

Aufgaben

1. Laden sie zunächst das Beispiel des Doppelklicklichtschalters (*lightswitch.xml*). Schauen Sie sich das Systemverhalten im Simulator an.
   Führen sie eine Prüfung der vorhandenen Queries aus. Schalten Sie nun die Option: *Diagnostic Trace → some* ein. Prüfen Sie die erste Anfrage erneut. Es wird der Pfad bis zur Erfüllung dieser Eigenschaft im Simulator angezeigt.

2. Danach ist eine Ampelschaltung für eine Fußgängerampel mit Drücker zu modellieren. Die Fußgängerampel wird nur grün, wenn der Drücker betätigt wird.
   Laden Sie die Datei *Fussgängerampel-Anfang.xml* und vervollständigen Sie die Lösung.



   Details:
   - Betätigt ein Passant den Drücker, so soll die Ampel der Autos nach Ablauf von 15 Zeiteinheiten (ZE) auf Gelb schalten.
   - Die Gelbphase dauert 5 ZE.
   - Die Fußgängerampel wird grün, 1 ZE nachdem die Autoampel rot geworden ist.
   - Die Fußgängerampel wird also immer 21 ZE nach dem Drücken grün.
   - Die Grünphase für Fußgänger soll 10 ZE dauern.
   - Die entsprechende Rotphase für Autos soll 12 ZE dauern (zur Sicherheit).

   Erklärung:
   Es sind bereits 3 Templates angelegt: *Fuss* – für die Fußgängerampel, *Autos* – für die Autoampel und *Passant* – dieses modelliert ein unendlich oft wiederholtes, zu beliebigem Zeitpunkt erfolgendes Betätigen des Drückers, d.h. es ist auch möglich das der Drücker nie betätigt wird, oder nur endlich oft, oder ganz oft hintereinander.

   Im Teil der globalen Deklarationen sind bereits 2 Channels deklariert. Der Channel *press* dient dazu dem *Fuss*-Prozess ein Betätigen des Drückers anzuzeigen. Die entsprechende Kante im *Fuss*-Template ist schon vorhanden. Der Channel *request* dient dazu, dass der *Fuss*-Prozess dem *Autos*-Prozess selbiges signalisiert. Dieser Vorgang ist ebenfalls schon modelliert.

   Die Ecke *Gedrückt* im Template *Fuss* ist vom Typ *urgent*, dies bedeutet, dass diese Ecke gleich wieder verlassen wird, wenn sie erreicht wird.

Die Ecke *GruenWarten* im Template *Autos* bedeutet, dass die Ampel noch Grün ist, sie aber nach 15 ZE auf Gelb schalten muss. Es existieren 2 Ecken für Gelb, eine zwischen Grün und Rot (*GelbG*) und eine zwischen Rot und Grün (*GelbR*).

Weiterhin sind schon alle Plätze modelliert. Die Templates *Fuss* und *Autos* besitzen jeweils eine lokale Uhr *x*. Bei diesen sind die fehlenden Kanten und deren Guard-, Synchronisations- und Updatelabels zu modellieren, sowie die Invarianten der Ecken *Rot, GelbG* und *GelbR.*

3. Vollziehen sie das Systemverhalten regelmäßig im Simulator nach, bis das System augenscheinlich korrekt funktioniert.

4. Formulieren und prüfen Sie die folgenden Eigenschaften:
   - Die Zustände *Fuss.Gruen, Autos.GelbG, Autos.GelbR* und *Autos.Rot* können erreicht werden (4 einzelne Queries)
   - Ist die Fußgängerampel grün, so ist in jedem Falle die Autoampel rot.
   - Wurde der Drücker betätigt, so wird die Fußgängerampel zwangsläufig grün.

5. Erweitern Sie die Aufgabenstellung um eine Straßenbahnschaltung. Dabei soll, wenn eine Bahn in die Nähe der Ampel fährt, diese noch 5 ZE länger grün zeigen, falls der Taster des Fußgängers gedrückt wurde. Damit kann die Bahn die Ampel noch vor der Rotphase passieren. Falls die Ampel sich schon in der Gelb- bzw. Rotphase befindet, soll durch die Bahnschaltung keine Änderung im Ampelverhalten stattfinden, d.h. die Rot- und Gelbphasen bleiben unberührt. Dabei ist es sinnvoll, ein weiteres Template analog zum Fußgänger für die Straßenbahn anzulegen.