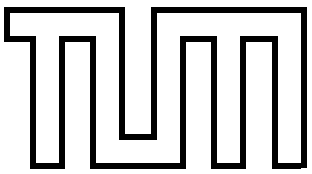

Technische Universität München

Fakultät für Informatik
Forschungs- und Lehreinheit Informatik VI

Übung zur Vorlesung Echtzeitsysteme

Aufgabe 4 – Esterel

Simon Barner
barner@in.tum.de

Irina Gaponova
gaponova@in.tum.de

Stephan Sommer
sommerst@in.tum.de

Wintersemester 2009/10

Aufgabe 2: Esterel

In der Vorlesung haben Sie die Programmiersprache *Esterel* kennengelernt. Das 3. Übungsblatt vertieft dieses Wissen anhand kleiner Aufgaben. Für das Bearbeiten der Aufgaben muss die Syntax von Esterel und der Esterel Studio Safe State Machine (SSM) Editor verwendet werden. Sowohl eine Kurzreferenz als auch eine vollständige Sprachbeschreibung zu Esterel finden Sie im Ordner `Uebung03` auf dem Laufwerk Q:.

Allgemeines

Für die Übungsaufgaben wird die Entwicklungsumgebung *Esterel Studio* von Esterel Technologies verwendet (*Start* → *Programme* → *Esterel Studio* → *Esterel Studio*). Wer sich außerhalb der Übung weiter mit Esterel beschäftigen will, findet unter `http://www-sop.inria.fr/esterel.org/files/Html/Downloads/Downloads.htm` einen kostenlosen Esterel-Compiler.

Safe State Machine – Modellierung

Zur Modellierung des aktuellen Aufgabenblattes benötigen Sie nur wenige grafische Elemente, die im SSM-Editor angezeigt werden. Darunter zählen das Kreissymbol für die Repräsentation eines Zustandes, die Verbindungslinie zum Verbinden von Zuständen, wobei die Linie dem Signal entspricht, das für diesen Zustandswechsel verantwortlich ist bzw. das emitiert wird, und den initial connector, der den Anfangszustand repräsentiert. Um die *input* und *output* Signale in der SSM festlegen zu können, müssen Sie wie in Abbildung 1 gezeigt vorgehen.

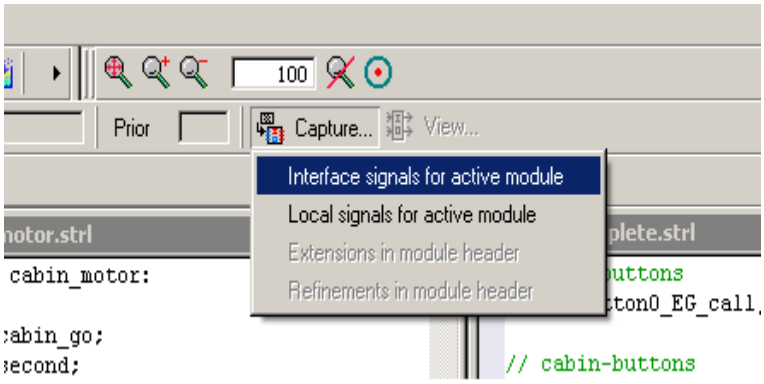

Abbildung 1: Setzen der Interfaces eines Moduls

Hinweis: Simulation

Ein Esterel-Programm kann mittels Simulation überprüft werden. Dazu sind folgende Schritte notwendig:

- Starten der Simulationsumgebung über den Menüpunke *Simulate* (siehe Abb. 2).
- Starten eines Simulationslaufs (siehe Abb. 3).

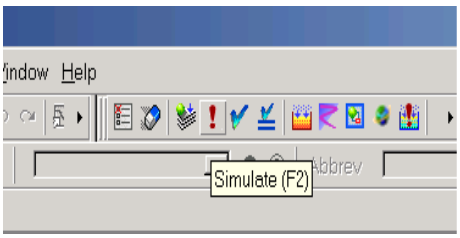

Abbildung 2: Starten der Simulationsumgebung

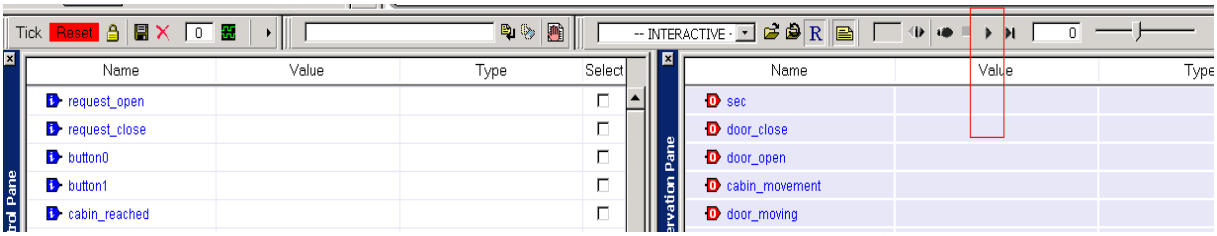

Abbildung 3: Simulationslauf beginnen

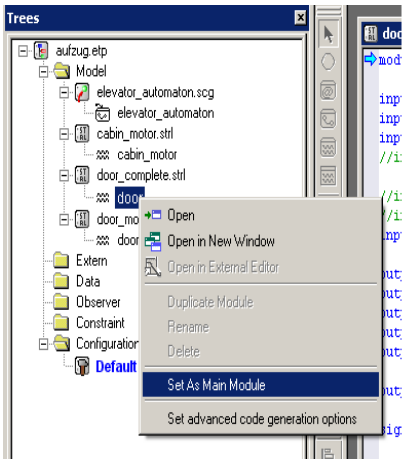

Abbildung 4: `door_complete.strl` als *Main Module* festlegen

Ansteuerung eines Aufzugs

Im Rahmen dieser Übung soll das bereits aus der Vorlesung bekannte Beispiel einer Aufzugsteuerung schrittweise erweitert werden. Estellen Sie zuerst ein Projektverzeichnis **(nicht auf Laufwerk Z:)** und kopieren Sie anschließend die folgenden Dateien in Ihr Projektverzeichnis:
`door_complete.strl`, `door_motor.strl`, `cabin_motor.strl`, `observer.strl` und `elevator_automaton.scg`. Erstellen Sie nun ein neues Esterel-Projekt und fügen die Dateien zum Projekt hinzu. Legen Sie die Datei `door_complete.strl` als *Main Module* fest (siehe Abbildung 4).

Nach Änderungen sollten Sie durch Simulation überprüfen, ob sich Ihr Programm noch richtig verhält.

Aufgabe: Erweiterter Aufzug

Ändern Sie das Programm / die SSM so ab, dass...

a) ...bei jedem *tick* das Signal *second* emitiert wird. Sehen Sie sich dazu den Programmcode, v.a. in `door_complete.strl` an und erweitern Sie das Programm entsprechend. Hinweis: Verwenden Sie dazu einen weiteren *loop*-Block.

b) ...der Aufzug über Taster für das Zielstockwerke verfügt (*goto_level0*, *goto_level1*).

c) ...der Aufzug bei Ankunft die Türen automatisch öffnet und vor Abfahrt automatisch schliesst (`cabin_motor.strl`). Verwenden Sie dazu die Signale `door_open` und `door_close`.

Überprüfen Sie nach jedem Schritt mittels Simulation die Funktionsfähigkeit Ihres Programms.

Aufgabe: Aufzug für 3 Stockwerke

Erweitern Sie ihr Programm für drei Stockwerke. Überlegen Sie sich, welche Signale Sie dafür benötigen. Beginnen sollten Sie mit der Erweiterung der SSM. Sie können sich an der SSM in Abbildung 5 orientieren.

Der Aufzug sollte folgende Eigenschaften haben:

- Für jedes Zielstockwerk existiert ein Taster.
- Der Aufzug darf einen Auftrag nach dem anderen abarbeiten.

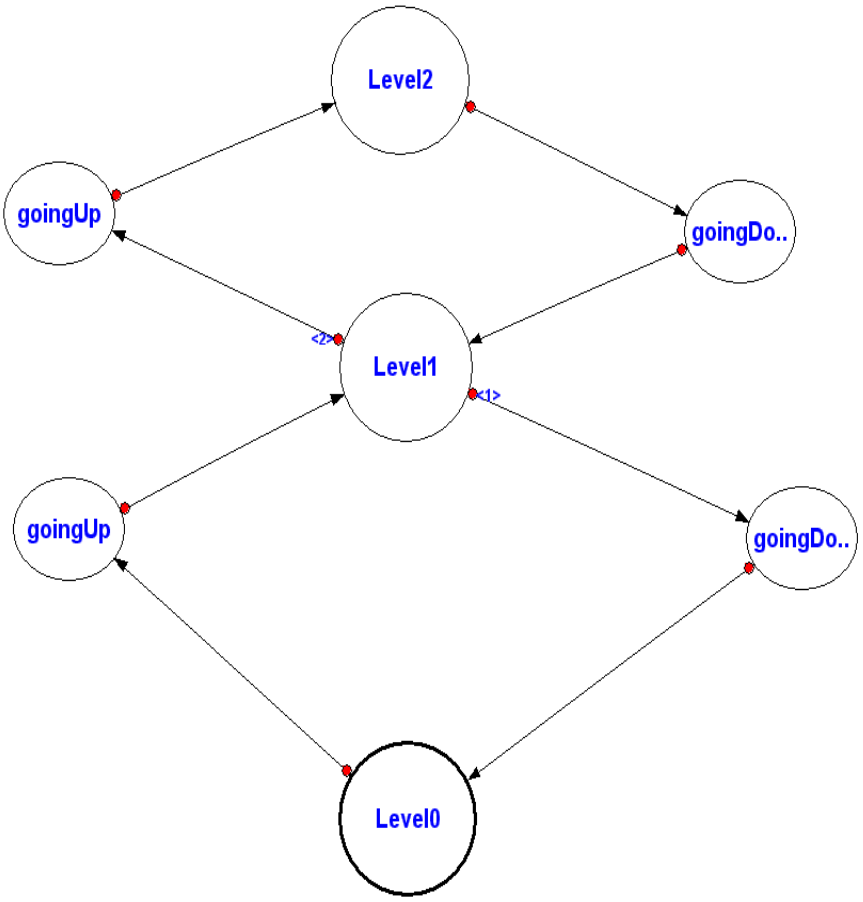

Abbildung 5: SSM für einen Aufzug mit 3 Stockwerken