Institut für Theoretische Informatik
Logik und Formale Methoden
Prof. Dr. P. H. Schmitt, C. Engel, D. Faragó, R. Krenický, M. Ulbrich, B. Weiß

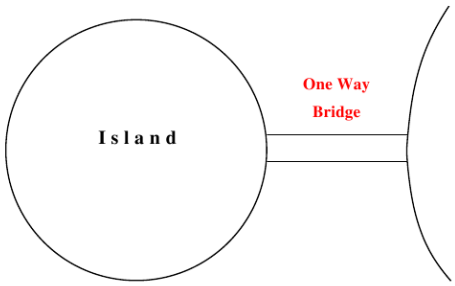
28.10.2009

Praktikum
Formale Entwicklung objektorientierter Software
Übungsblatt 1: RODIN und Event-B

Aufgabe 1 — Umgebung

Loggen Sie sich mit Ihrem Praktikumskonto auf einem Institutsrechner ein und ändern Sie das Passwort (mit dem Befehl `passwd`). Die Rechner stehen im Poolraum 305 und sind von außen über ssh erreichbar (z.B. `i11pc50.ira.uka.de`).

Die Rechner sind so konfiguriert, dass Sie RODIN mit dem Befehl `runrodin` von der Kommandozeile aus starten können. Starten Sie das Programm und finden Sie sich mit der Oberfläche zurecht.



Aufgabe 2 — "Die Brücke"

Auf der Praktikumshomepage (Adresse s.u.) finden Sie einen Link zu einem Tutorium für RODIN, in dem ein sehr einfaches Modell für eine einspurige Brücke zwischen einer Insel und dem Festland in mehreren Verfeinerungsstufen besprochen wird. Gegenstand der Verfeinerungen sind dabei:

Ausgangsmodell *M0* Modellierung von kapazitätsbeschränkter Insel und Festland

Erste Verfeinerung *M1* Einführung der Brücke als Einbahnstraße

Zweite Verfeinerung *M2* Einführung von Ampeln zur Beschränkung des Verkehrs

Dritte Verfeinerung *M3* Einführung von Fahrbahn-Kontakten zur Steuerung der Ampelanlage.

In der vorinstallierten RODIN-Umgebung finden Sie das Projekt "bridge". **Aufgabe: Öffnen Sie es und erarbeiten Sie sich anhand der Tutoriumsbeschreibung die Maschinen *M0* bis *M2***. Vorerst müssen keine Modelle selbst erstellen, Sie sollten aber das generelle Vorgehen verstehen. Maschine *M3* ist etwas komplizierter und technischer. Diese Maschine und ihre Verfeinerungs-Idee müssen Sie nicht unbedingt in aller Tiefe nachvollziehen.

In den Unterlagen zur Vorlesung finden Sie die benötigte Theorie zu den Beweisverpflichtungen und zu den grundsätzlichen Ideen der Verfeinerung. Zusätzlich liegt auf der Seite

zum Praktikum eine Übersicht über die auftretenden Beweisverpflichtungen zum Herunterladen bereit, aus der Sie erkennen können, wie ausgehend von Kontexten und Maschinen Beweisverpflichtungen entstehen.

In dem Projekt, das Sie geöffnet haben, sind die Beweisverpflichtungen für die Maschinen noch nicht bewiesen. **Aufgabe: Benutzen Sie die eingebauten automatischen Beweiser, um noch offene Beweisverpflichtungen zu schließen.**

Die Beweise zu Maschine *M0* können in der vorliegenden Form nicht alle geschlossen werden. Die Modellierung entspricht der auf Seite 4 des Tutoriums vorgestellten ersten Fassung. Sie müssen zunächst die auf Seite 5 oben beschrieben Änderungen in das Modell einarbeiten, um alle Beweise schließen zu können.

Hinweise zum Tutorium

Die Dokumentation weicht etwas von dem Projekt ab, das Sie in Ihrem Arbeitsbereich finden, die Unterschiede sind aber minimal:

- Die Beweisverpflichtungen, die durch die Regel FIS aufgeworfen werden, sind trivial und werden von RODIN gar nicht erst erzeugt. Wenn eine Zuweisung deterministisch ist, ist ihre Machbarkeit offensichtlich selbstverständlich.
- Die Eigenschaften prp_0_1 und prp_0_2 finden sich im Kontext Env wieder.
- Die Beweisverpflichtung für die Unmöglichkeit von Verklemmungen (Deadlock freeness) wird nicht automatisch erzeugt sondern, von Hand als Theorem DLKF in den einzelnen Maschinen hinzugenommen.
- Die Beweisverpflichtungen, die für die Verfeinerung erzeugt werden, tragen das Suffix /GRD
- Die Ampel in *M2* ist nicht mit 0 und 1 modelliert sondern mit einer Trägermenge lights und den Konstanten GREEN und RED. Die sich ergebenden Änderungen sind offensichtlich.

Beweis-Werkzeuge

Folgende Werkzeuge sind in RODIN integriert und können benutzt werden, um eine Verpflichtung zu entlasten:

- *Retry Auto Provers.* Diesen Befehl finden Sie im Kontextmenü des *Event-B Explorers* für jeden Beweisknoten oder Beweisgruppenknoten. Es handelt sich dabei um einen recht schwachen Beweiser, der aber den Großteil der entstehenden Verpflichtungen schon zeigen kann.
- *M0, M1, M2, M3, ML.* Diese speziellen Beweiser finden Sie, wenn Sie eine konkrete Beweisverpflichtung öffnen im Fenster *Proof Control*. Es heißt:

  > ML applies a mix of forward, backward and rewriting rules in order to discharge the goal [...].

  Der Beweiser versucht also allgemeine Regeln der Mengenlehre auf die Verpflichtung anzuwenden und diese so zu zeigen.
- *P0, P1, PP* Diese drei Ausbaustufen des *PP*-Beweiser (im selben Fenster wählbar) gehen anders vor. Alle Ausdrücke werden in Prädikatenlogik erster Stufe übersetzt. Als Primitivum bleiben dabei einzig das Prädikat $\in$ und die map-Funktion $\mapsto$ erhalten. Das Resultat wird an einen prädikatenlogischen Beweiser geschickt, der nichts von Mengenlehre verstehen muss. *PP* unterstützt aber auch arithmetische Ausdrücke (z.B. $+, *, <$)

- *NewPP* Eine neuere Implementierung, die nach dem selben Prinzip verfährt wie *PP*, aber keine arithmetischen Ausdrücke unterstützt. Dieser Beweiser wird über einen eigenen Knopf zur Linken von ML/PP gestartet.

Die Ziffern im Namen (z.B. *P1* oder *M2*) stehen für die Mächtigkeit des Beweisers. Je höher diese Ziffer ist, desto mehr Formeln aus dem Umfeld werden für den Beweis herangezogen. Damit wird einerseits die Möglichkeit, einen Beweis zu finden, aber gleichzeitig die Wahrscheinlichkeit auf eine falsche Fährte zu gelangen oder zu viel Zeit für einen einfachen Beweis zu benötigen, erhöht.

Für weitere Informationen über die eingesetzten automatischen Beweiser können Sie die Seite `http://wiki.event-b.org/index.php/Rodin_Provers` konsultieren.

Aufgabe 3 — Die "Zugbrücke"

In dieser Aufgabe sollen Sie selbst ein Verfeinerung einer bestehenden Maschine erstellen und die entsprechenden Beweisverpflichtungen beweisen. Dazu wollen wir das Modell der Brücke, die Insel und Festland verbindet, zu einer Zugbrücke (oder besser "Hebebrücke", s. Abb.) hin erweitern.

Die Brücke wird dazu mit einer Hebevorrichtung ausgestattet, mit der die Fahrbahn angehoben werden kann, damit Schiffe und Boote mit höheren Aufbauten darunter passieren können.

Die Brücke ist eine Zugbrücke mit zwei dedizierten Zuständen: oben und unten.	EQP-6

Offensichtlich sorgt die Hebemechanik dafür, dass die Brücke nicht befahren werden kann, wenn die Fahrbahn gehoben ist. Das ist eine Eigenschaft, die sich aus der Natur der Anlage ergibt.

Solange die Brücke angehoben ist, können keine Autos auf die Brücke auffahren.	EQP-7

Die Anlange könnte sich – rein technisch – natürlich heben, auch wenn ein Auto auf der Fahrbahn ist, aber wir wollen als Funktionalitätsanforderung dieses ausschließen.

Die Brücke kann nur angehoben werden, wenn keine Autos auf der Brücke sind.	FUN-4

Einführung der Zugbrücke

Erstellen Sie im bestehenden Projekt "bridge" eine neue Maschine *M1_ZB1* als Verfeinerung der Maschine *M1*. Am einfachsten geht das, indem Sie mit der rechten Maustaste im Rodin-Explorer auf *M1* klicken und aus dem Kontextmenü *REFINE* wählen.

Die damit angelegte Maschine ist als verfeinernde Kopie von *M1* ausgelegt: Sie kopiert alle Ereignisse der abstrakteren Maschine und dupliziert ebenfalls alle Variablen aus der Maschine höherer Abstraktion.

Fügen Sie nun zu diesem Modell eine neue Variable hinzu, die den Zustand der Brücke beschreibt. Fügen Sie außerdem zwei neue Invarianten zu der Maschine hinzu, die die Anforderungen EQP-6, EQP-7 und FUN-4 abbilden.

Ergänzen Sie die bestehenden Ereignisse um die den Anforderungen entsprechenden Vorbedingungen (guards) und evtl. Zuweisungen. Erstellen Sie ebenso zwei neue Ereignisse *BRIDGE_UP* und *BRIDGE_DOWN*, die das Heben und Senken der Brücke modellieren.

Beweisen Sie die zur Maschine gehörigen Beweisverpflichtungen.

Konvergenz

Für die Maschine `M1_ZB1` ist folgender unendliche Run(?) möglich:

INITIALISATION, *BRIDGE_UP*, *BRIDGE_DOWN*, *BRIDGE_UP*, *BRIDGE_DOWN*, ...

Das bedeutet, dass aus Sicht der Maschine `M1`, die von "up" und "down" ja nichts weiß, nach der Initialisierung kein weiteres Event (als Verfeinerung) jemals wieder auftreten kann. Man sagt in diesem Fall, dass der Run *divergiert*.

Wir wollen sicherstellen, dass jeder Run *konvergiert*: Zwischen je zwei Ereignissen, die ein Ereignis einer abstrakteren Maschine verfeinern, dürfen dann höchstens endlich viele neu eingeführte Ereignisse auftreten.

Dazu definieren wir eine *Variante*: Einen Term, der eine natürliche Zahl definiert, und der durch jedes neu eingeführte Ereignis echt kleiner gemacht wird. Da es keine unendlich absteigenden Ketten in den natürlichen Zahlen gibt, ist damit die Konvergenz sichergestellt.

Die konvergente Zugbrücke

Da wir in diesem Beispiel keine Schiffe modellieren, aber den oben stehenden Run ausschließen (*Starvation*!) wollen, müssen wir die folgende zusätzliche Anforderung aufnehmen:

Zwischen zwei aufeinander folgenden Hebevorgängen der Brücke muss wenigstens ein Auto die Brücke passieren.	FUN-5

Ändern Sie den Typ der neuen Ereignisse in *M1_ZB1* von "ordinary" auf "anticipated", um anzuzeigen, dass diese Ereignisse in einer späteren Verfeinerung konvergent sein werden.

Erstellen Sie eine neue Maschine *M1_ZB2* als Verfeinerung von *M1_ZB1*. Markieren Sie hier die beiden interessanten Ereignisse als "convergent". Fügen Sie eine Zählvariable hinzu und passen Sie die Ereignisse an, um die in FUN-5 geforderte Eigenschaft zu erfüllen.

Finden Sie eine geeignete Variante, die sich des Brückenzustandes und der Zählvariable bedient und die sowohl durch *BRIDGE_UP* als auch durch *BRIDGE_DOWN* echt kleiner gemacht wird.

Beweisen Sie alle aufgekommenen Beweisverpflichtungen.

Bitte beachten Sie für diese und die folgenden Aufgaben:

Geben Sie in den Kommentaren zu Invarianten, Variablen, Theoremen, Ereignissen, Guards etc. diejenige Anforderung an, die dieses Element bedingen. Jede Anforderung sollte sich in wenigstens einem Element wiederfinden lassen.

Abgabe am 9.11.2009

Die Abgabe erfolgt durch Vorführen während des Praktikumtermins. Bitte dokumentieren Sie Ihre Maschinen und Kontexte ausreichend und seien Sie darauf vorbereitet, sie in der Abnahme zu erklären.

Praktikums-Webseite: `http://lfm.iti.kit.edu/keyprakt0910.php`

Christian Engel:	R. 106,	Tel. 608-4338,	E-Mail: `engelc@ira.uka.de`
David Faragó:	R. 308,	Tel. 608-7322,	E-Mail: `farago@ira.uka.de`
Roman Krenický:			E-Mail: `krenicky@ira.uka.de`
Mattias Ulbrich:	R. 106,	Tel. 608-4338,	E-Mail: `mulbrich@ira.uka.de`
Benjamin Weiß:	R. 309,	Tel. 608-4324,	E-Mail: `bweiss@ira.uka.de`