Hinweise zur Arbeit mit UPPAAL

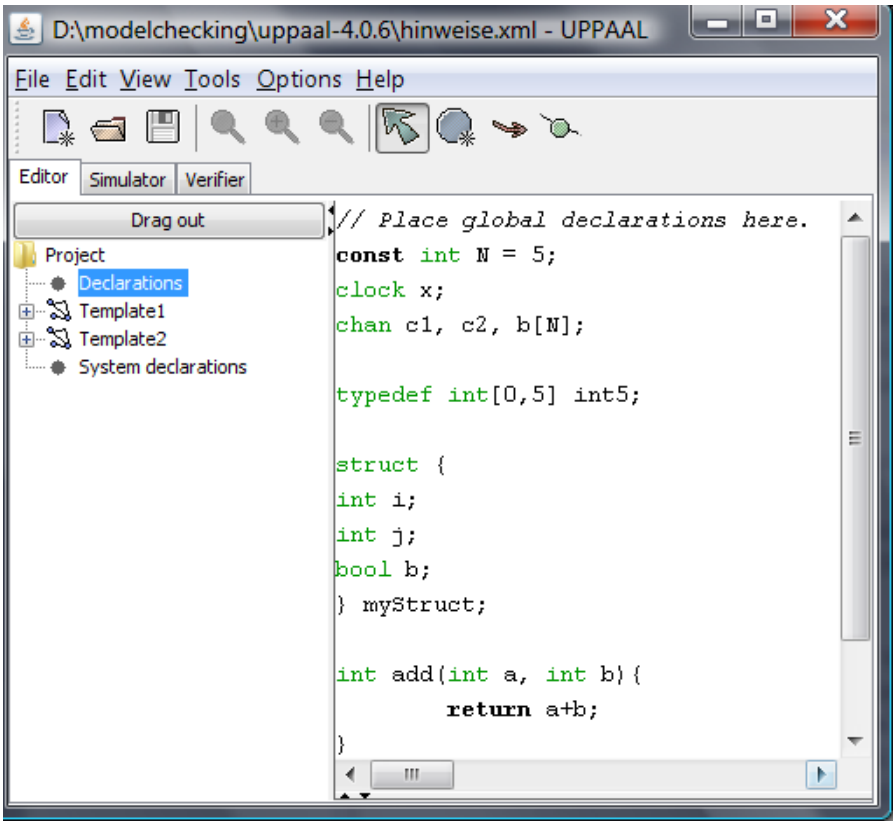
Deklarationen

Nach dem Öffnen des Tools befindet man sich zunächst im Hauptbildschirm. Auf der linken Seite befindet sich die Übersicht über die einzelnen Komponenten des Systems. Unter „Declarations“ lassen sich globale Variablen und Channels deklarieren, sowie Konstanten, Typen und Funktionen definieren. Integern kann ein Wertebereich zugewiesen werden, so besagte z. B. die Deklaration

```
int[0,5] i;
```

Dass `i` nur Werte von 0 bis 5 annehmen kann.

Jedes Template besitzt zudem einen eigenen Deklarationsbereich, in dem lokale Variablen usw. deklariert und definiert werden können.



Unter „System declarations“ wird die Komposition des Systems bestimmt. Soll jedes Template nur einmal instanziiert werden, so genügt es die Namen dieser hinter dem Schlüsselwort **system** aufzulisten:

```
system Template1, Template2;
```

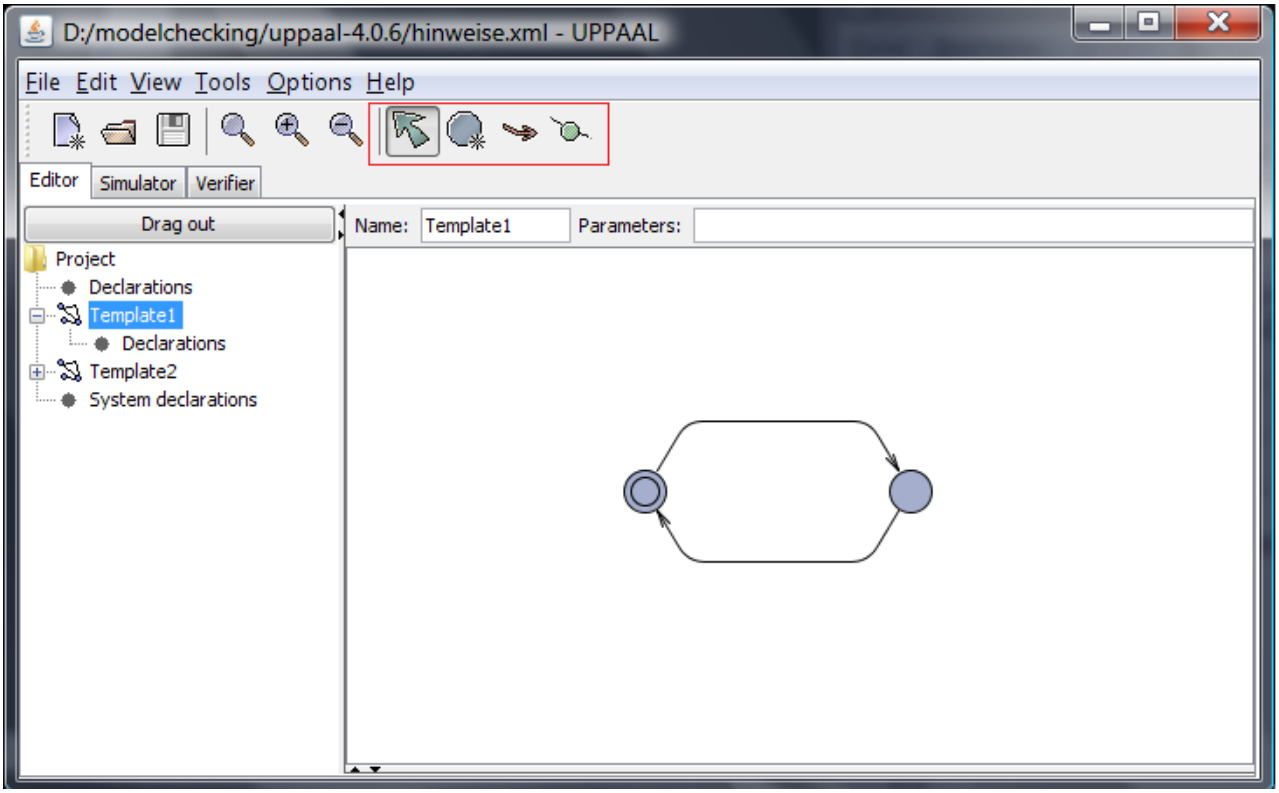
Sollen einige jedoch mehrmals, oder mit bestimmten Parametern instanziiert werden so muss dies explizit geschehen, z. B.:

```
ta  = TemplateA(5,true);
tb1 = TemplateB();
tb2 = TemplateB;
system ta,tb1,tb2;
```

Graphischer Editor

Timed Automata werden in UPPAAL als gerichtete Graphen gezeichnet.

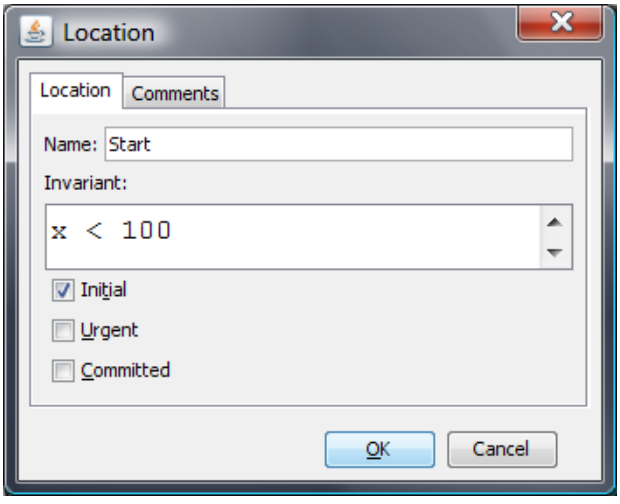
In der Werkzeugleiste gibt es 4 Werkzeuge zur Auswahl:



Mit dem ersten kann jedes vorhandene Objekt bearbeitet werden, mit dem zweiten werden neue Ecken des Graphen erstellt, mit dem Dritten neue Kanten und mit dem vierten können Kanten neue Knickpunkte hinzugefügt werden.

Ecken

Wird mit dem Pfeil-Werkzeug doppelt auf eine Ecke geklickt, so öffnet sich ein Fenster, in dem der Name der Ecke und die Invariante angegeben werden können. Zudem kann ausgewählt werden, ob die Ecke die initiale ist und ob sie dringend (*urgent*) oder verpflichtend (*committed*) ist.



Die **Invariante** muss ein Seiteneffektfreier Ausdruck sein, der zu wahr oder falsch ausgewertet werden kann.

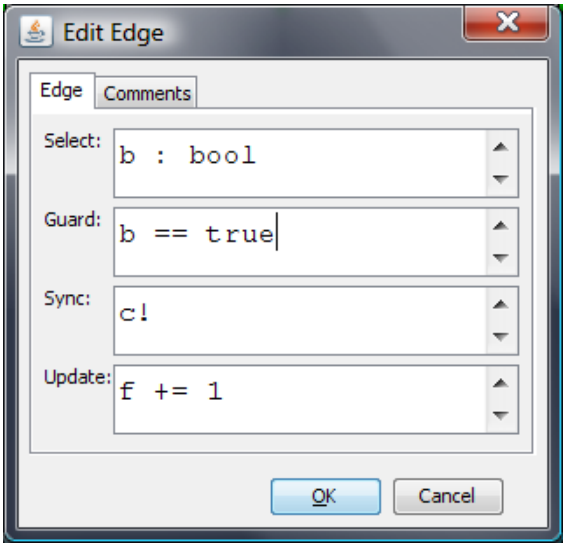
Solange sich der Automat an dieser Ecke befindet, darf diese Invariante nicht verletzt werden. Zustandsübergänge die eine Invariante verletzen würden, werden nicht ausgeführt.

Eine **dringende** Ecke (*urgent*) muss, wenn erreicht, ohne Zeitverzögerung wieder verlassen werden. Dazu ist analog eine neue Uhr `x` einzuführen, welche auf allen eingehenden Kanten auf Null gesetzt wird sowie die Invariante `x==0`. Enthält ein Systemzustand mehrere dringende Ecken, so wird nichtdeterministisch eine ausgewählt, welche im nächsten Zustand verlassen wird.

Eine **verpflichtende** Ecke (*committed*) ist noch restriktiver. Enthält ein Systemzustand eine verpflichtende Ecke, so muss diese im nächsten Zustandsübergang verlassen werden. Eine verpflichtende geht also vor einer dringenden Ecke. Auch hier gilt wieder: sind mehrere verpflichtende Ecken aktiv, so wird nichtdeterministisch eine ausgewählt welche im nächsten Zustand verlassen wird.

Kanten

Wird doppelt auf eine Kante geklickt erscheint ein Fenster, in dem Select-, Guard-, Synchronisations- und Update-Labels spezifiziert werden können.



Select-Labels weisen einem Wert nichtdeterministisch einen Bezeichner aus einem bestimmten Wertebereich zu, der in den anderen Labels verwendet werden kann. Das Select-Label

```
z : int[0,3]
```

zum Beispiel weise dem Bezeichner `z` den Wert 0, 1, 2 oder 3 zu.

Guard-Labels beschreiben, unter welchen Bedingungen eine Kante schalten kann. Sie müssen wie Invarianten seiteneffektfrei sein (dürfen also z. B. keine Variablenzuweisungen enthalten) und müssen zu wahr oder falsch ausgewertet werden können.

Ein Guard kann einfache Bedingungen an Uhren beinhalten, Unterschiede zwischen Uhren und Ausdrücke, die keine Uhren beinhalten.

Beispiele:

- x < y

  x muss kleiner als y sein
- `(clock1-clock2) > 10`

  Die Differenz zwischen `clock1` und `clock2` muss größer als 10 sein

Synchronisations-Labels dienen zur Synchronisation der Prozesse untereinander über Channels. Sie haben die Form `e!` oder `e?`, wobei `e` ein Channel ist, oder ein Ausdruck der zu einem Channel ausgewertet werden kann. Die Kante mit dem Label mit Ausrufezeichen ist dabei der Initiator der Synchronisation, die mit Fragezeichen der Empfänger.

Zwei Prozesse können sich über Kanten synchronisieren, welche aktiv sind (Guards erfüllt) und komplementäre Synchronisations-Labels besitzen. Synchronisation bedeutet, dass beide Kanten zur selben Zeit schalten, d. h. die aktive Ecke beider Prozesse ändert sich auch gleichzeitig.

Weiterhin gibt es *urgent channels*, also dringende. Bei einem dringenden Channel ist es erforderlich die Synchronisation auszuführen wenn sie möglich ist. Bei einem regulären Channel wäre es auch erlaubt andere mögliche Zustandsübergänge vorzuziehen. Guards, in denen Uhren vorkommen sind nicht an Kanten erlaubt, welche ein Synchronisations-Label haben, dass sich auf einen dringenen Channel bezieht.

Eine zusätzliche Erweiterung sind *broadcast channels*. Diese erlauben eine 1-n Synchronisation. Eine Kante mit dem Labe `b!`, wobei `b` ein Broadcast-Channel ist, kann immer feuern, egal ob eine aktive Kante mit komplementärem Label vorhanden ist. Jede aktive Kante mit solchem Label ist jedoch gezwungen sich zu synchronisieren. Für Guards besteht hier dieselbe Einschränkung wie für dringende Channels.

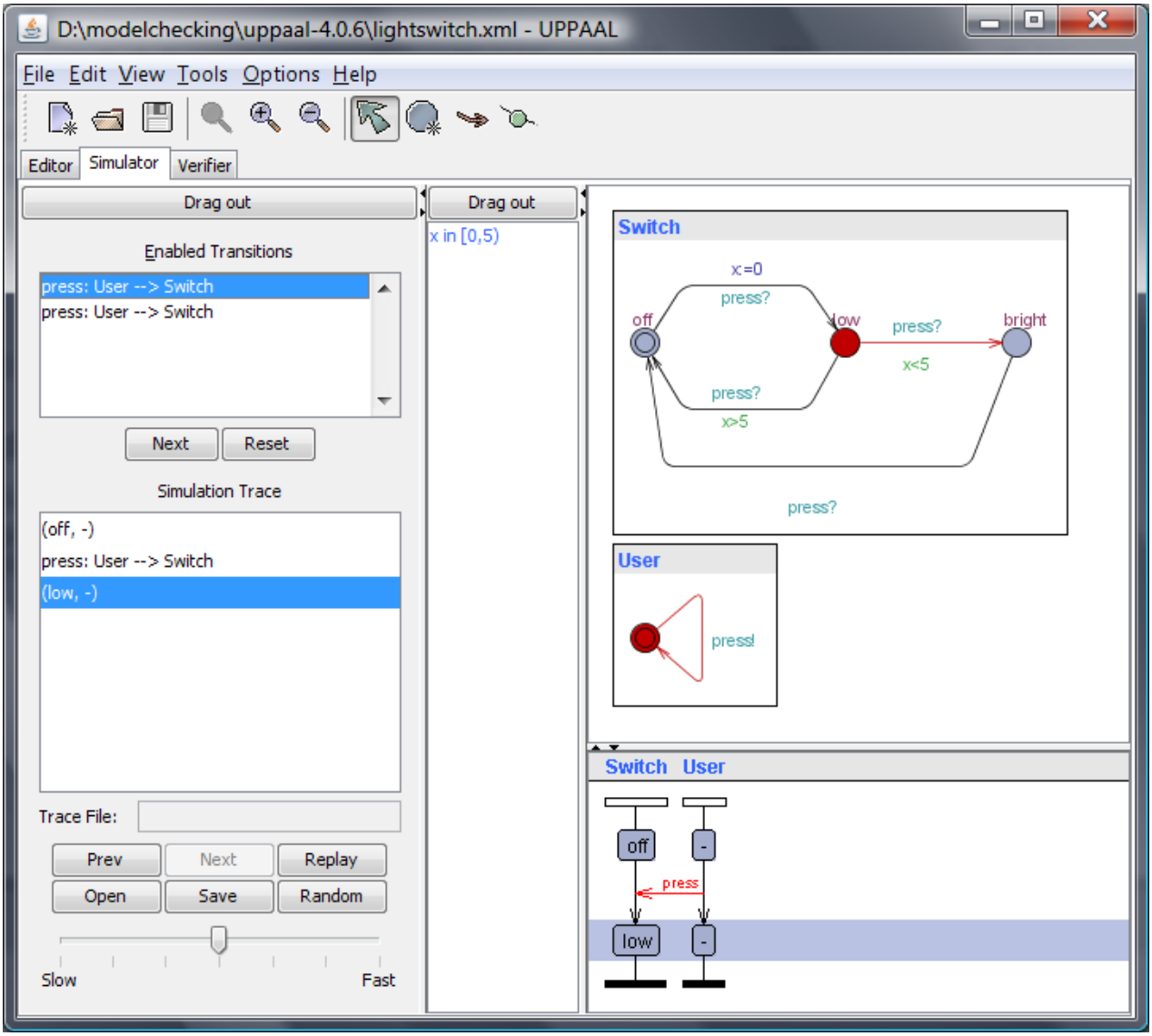
Update-Labels werden verwendet um Variablen oder Uhren neue Werte zuzuweisen. Es können ebenfalls Funktionen ausgeführt werden. Mehrere Update-Anweisungen können durch Komma getrennt beschrieben werden.

Beispiele:

- `x = 0`
- `x = 1, y = 2*x, myFunction(z)`

Update-Anweisungen werden in der Reihenfolge ausgeführt, in der sie stehen.

Simulator



Im Simulator können Ausführungen des Systems simuliert werden; zufällige, vom Nutzer gesteuerte und solche die zur Verletzung oder Erfüllung einer überprüften Systemeigenschaft führen.

Im linken oberen Fenster sieht der Nutzer die im aktuellen Zustand aktiven (gerade möglichen) Zustandsübergänge. Davon kann einer ausgewählt werden um ihn mittels des Next-Buttons auszuführen. Dieses Fenster ist für eine Schritt-für-Schritt Simulation gedacht.

Im linken unteren Fenster wird der Simulationspfad angezeigt, jeder Zustand kann im Nachhinein erneut betrachtet werden. Darunter befinden sich weitere Buttons. Mittels des Random-Buttons wird eine zufällige automatische Simulation gestartet, die aus den jeweils möglichen Transitionen zufällig eine auswählt. Die Simulation läuft mit der Geschwindigkeit ab, die mittels dem sich darunter befindlichem Schieberegler gewählt werden kann. Der Pfad kann in einer Datei gespeichert werden.

Im mittleren Fenster werden die aktuellen Werte aller Variablen und Uhren angezeigt. Bei Uhren wird meist kein genauer Wert angezeigt, sondern ein Intervall oder eine größer-als Angabe.

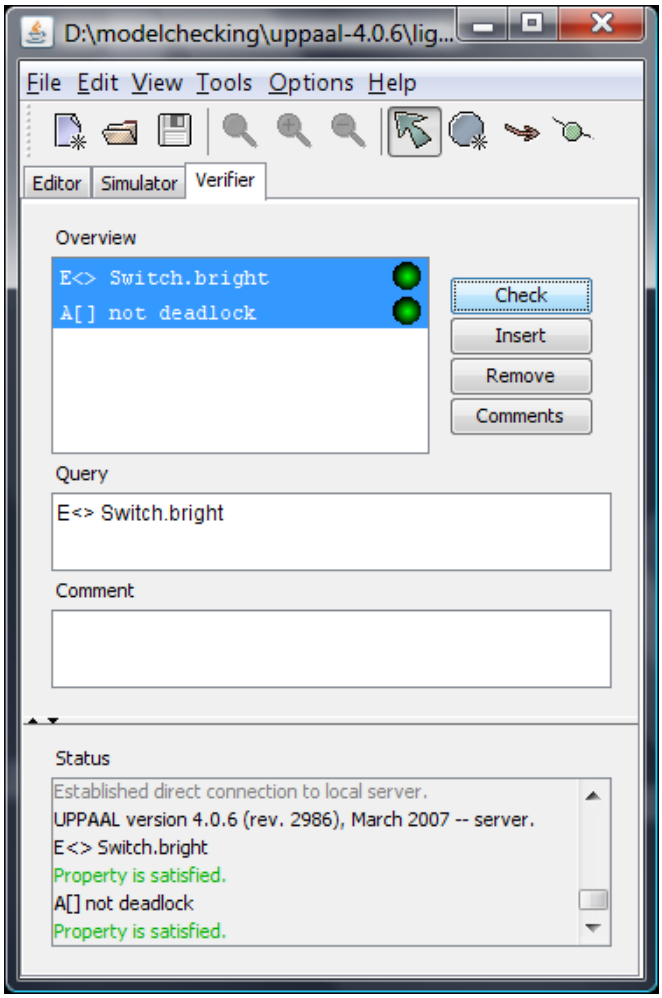
Das rechte obere Fenster zeigt die Automaten und deren aktive Ecken und Kanten graphisch an. Eine rot markierte Ecke beschreibt die aktuell aktive. Abhängig davon, welche Transition im linken oberen Fenster ausgewählt ist, können ein oder mehrere Kanten rot markiert sein. Diese würden im nächsten Schritt schalten.

Dort kann über ein Rechtsklick-Menü an Prozesse herangezoomt werden und es können EPS-Dateien erzeugt werden (Encapsulated Postscript).

Das rechte untere Fenster zeigt einen Message-Sequence-Chart des generierten Pfades an.

Model-Checker

Unter der Registerkarte „Verifier“ können Systemeigenschaften in Form von Queries formuliert werden.



Im Feld „Query“ kann die jeweils ausgewählte Eigenschaft bearbeitet werden. Mittels der Insert- und Remove-Buttons können neue hinzugefügt oder existierende gelöscht werden.

Der Check-Button für eine Modellprüfung der ausgewählten Query aus. Mittels der Shift- oder Strg-Taste können auch mehrere Queries auf einmal angewählt werden.

In der Hilfe findet sich eine ausführliche Beschreibung zur Syntax der Queries.

Zur Erinnerung wird hier aber noch einmal ein Überblick über die häufigsten Eigenschaften gegeben:

Möglichkeit

Für eine Aussage `p` beschreibt die Formel `E<> p`, dass ein Zustand existiert in dem `p` gilt. (Es existiert ein Pfad (`E`) auf dem es irgendwo einen Zustand gibt (`<>`) in dem `p` gilt).

Globale Invariante

Die Formel `A[] p` gilt dann, wenn in jedem Zustand die Aussage `p` gilt. (Auf allen Pfaden (`A`) gilt in allen Zuständen (`[]`) `p`).

Die äquivalente Formel ist `not E<> not p`. (Es gibt keinen Pfad (`not E`) auf dem es irgendwo einen Zustand gibt (`<>`) auf dem p nicht gilt (`not p`)).

Tritt zwangsläufig ein

Durch die Formel `A<> p` wird ausgesagt, dass unter allen Umständen ein Zustand erreicht wird, in dem `p` wahr ist. (Auf allen Pfaden (`A`) existiert irgendwo ein Zustand (`<>`) in dem `p` gilt).

Implikation

Der Ausdruck `p imply q` bedeutet: wenn `p` gilt, dann gilt zwangsläufig auch `q` (zur selben Zeit).

Führt zu

Der Ausdruck `p --> q` beschreibt, dass wenn p wahr wird, in der Zukunft auch q wahr werden muss, kurz `p` führt zu `q`.

Dieser Ausdruck ist äquivalent zu der Formel `A[] (p imply A<> q)`. (Auf allen Pfaden und in allen Zuständen gilt (`A[]`): wenn `p` gilt dann gilt auch folgendes: auf allen Pfaden existiert mindestens ein Zustand (`A<>`) in dem `q` gilt).

In UPPAAL werden jedoch keine verschachtelten Operatoren erlaubt, weshalb der Operator `-->` eingeführt wurde.

Deadlocks

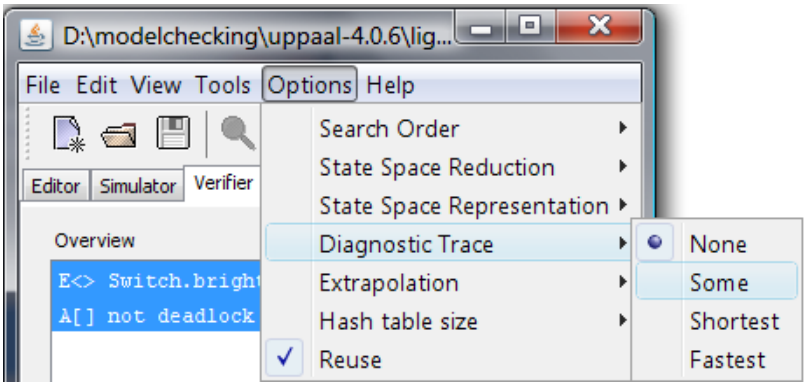
Der Ausdruck `deadlock` ist wahr in einem Zustand der keine Nachfolgezustände hat.

Übersicht der Eigenschaften:

Kann eintreten	`E<> p`
Gilt immer	`A[] p`
Wird eintreten	`A<> p`
Implikation	`p imply q`
Führt zu	`p --> q`
Keine Deadlocks	`A[] not deadlock`

Diagnostic Traces

Wird die Option Diagnostic Trace ausgewählt, so können Pfade, die zur Verletzung oder Erfüllung einer Systemeigenschaft führen, im Simulator nachvollzogen werden. Es gibt die Option irgendeinen Pfad zu berechnen (am schnellsten), den kürzesten, also den, der am wenigsten Zustandsübergänge beinhaltet, oder den schnellsten, also den, bei dem am wenigsten Zeit verstreicht.



Bei Eigenschaften der Form `E<> p` und `p --> q`, wird dann ein Pfad erstellt wenn die Prüfung ergibt, dass die Eigenschaft erfüllt ist. Dieser wird dann in den Simulator geladen, nachdem vom Nutzer bestätigt wird, dass der alte Pfad überschrieben werden darf.

Bei Eigenschaften, die besagen dass etwas nicht eintreten darf, wird dann ein Pfad erstellt, wenn die Prüfung ergibt, dass die Eigenschaft verletzt ist. Eine solche Eigenschaft ist z. B. `A[] not deadlock`.

Ist diese Option eingeschaltet, so kann immer nur eine Query auf einmal geprüft werden.