Gruppe: _____

Name: _____

Matr.Nr.: _____

Erfolg: _____

Übung 7

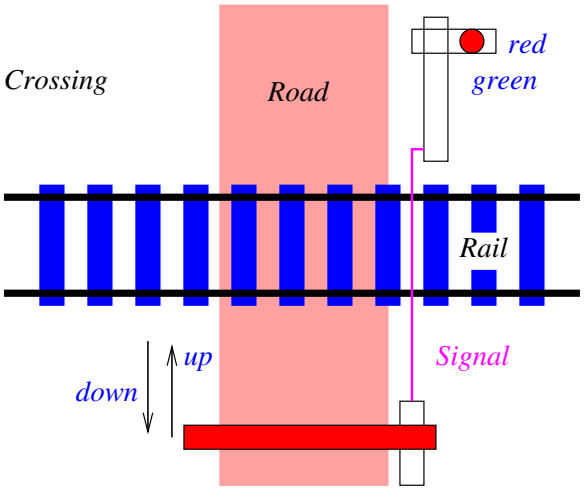
**Systemtheorie 1**

Wintersemester 2006/2007

Abgabe 14.12.2006 10:30

Institut für Formale Modelle und Verifikation, Dr. Toni Jussila, Dipl.-Ing. Robert Brummayer

## Einleitung




Gegeben sei der obige Bahnübergang, den Sie bereits aus FG3 kennen sollten. Wir betrachten jedoch eine leicht modifizierte Version. Der Bahnübergang kann durch die drei parallel laufenden Prozesse `Road`, `Rail` und `Signal` modelliert werden, die wie folgt in Prozeß-Algebra-Notation definiert sind:

```
Road   = car.up.center.cexit.down.Road
Rail   = train.green.tenter.texit.red.Rail
Signal = green.red.Signal + up.down.Signal
```

Die Ereignisse haben dabei folgende Semantik:

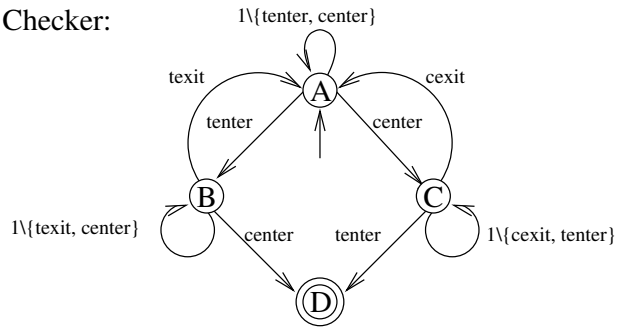
- `car` bzw. `train` bedeutet, dass ein Auto bzw. Zug den Bahnübergang erreicht hat.
- `center` bzw. `tenter` bedeutet, dass ein Auto bzw. Zug begonnen hat den Bahnübergang zu überqueren.
- `cexit` bzw. `texit` bedeutet, dass ein Auto bzw. Zug den Bahnübergang erfolgreich überquert hat.

- `green` bzw. `red` bedeutet, dass die Ampel des Zuges auf grün bzw. rot geschaltet wurde.
- `up` bzw. `down` bedeutet, dass die Schranke geöffnet bzw. geschlossen wurde.

Wie man erkennen kann, ist ein Unfall genau dann möglich, wenn ein Auto mit der Überquerung begonnen hat und ein Zug mit der Überquerung beginnt, bevor das Auto die Überquerung beendet hat und vice versa. In Prozeß-Algebra-Notation entsprechen die Folgen `center.tenter` bzw. `tenter.center` diesem Unfall, den es zu vermeiden gilt.

Der folgende Checker-Automat dient der Verifizierung, ob ein Unfall auftreten kann oder nicht. Das Symbol `1` steht dabei für die Menge aller möglichen, vorher diskutierten, Ereignisse:

Checker:




**Aufgabe 25**

a) Erstellen Sie für `FSMCalc` die drei Dateien `road.lts`, `rail.lts` und `signal.lts` für die in der Einleitung definierten Prozesse. Drucken Sie die drei Dateien aus und geben Sie den Ausdruck mit ab.

b) Erstellen Sie für `FSMCalc` die Datei `checker.fsm` für den in der Einleitung definierten Checker-Automaten. Drucken Sie die Datei aus und geben Sie den Ausdruck mit ab.

c) Verifizieren Sie mit `FSMCalc`, ob ein Unfall auftreten kann oder nicht. Rufen Sie `FSMCalc` mit der Operation `poc` auf. Übergeben Sie dabei als Argumente zuerst den in `b)` erstellten Checker-Automaten (mit -fsm) und danach die drei in `a)` erstellten LTS (jeweils mit -lts). Drucken Sie das Ergebnis von `FSMCalc` aus und geben Sie den Ausdruck mit ab. Interpretieren Sie schließlich das Ergebnis. Kann ein Unfall auftreten oder nicht? Begründen Sie Ihre Aussage.

Hinweis 1: Die Operation `poc` von `FSMCalc` schaltet die übergebenen LTS parallel. Die Semantik dieser Parallelschaltung durch Interleaving ist die des Parallel-Operators aus der Prozeß-Algebra, wie wir sie in FG3 kennengelernt haben. Der Checker-Automat wird per Produkt-Automaten-Konstruktion mit den parallel geschalteten LTS kombiniert.

Hinweis 2: Die Operation `poc` verwendet die Partial-Order-Reduction, die wir später noch kennenlernen werden und die für die Lösung dieses Beispiels keine Rolle spielt.

Hinweis 3: Warnungen von FSMCalc, dass der Checker nicht invariant gegenüber einiger lokalen Transitionen sei, können Sie ignorieren. Diese Warnungen treten auch bei einer korrekten Lösung auf.

Hinweis 4: `FSMCalc` interpretiert ein LTS als Automaten, bei dem alle Zustände End-Zustände sind.

Hinweis 5: Ob ein Automat einen erreichbaren Finalzustand hat, können Sie mit Hilfe der Operation `check` von `FSMCalc` überprüfen.

**Aufgabe 26**

Wir modifizieren nun den Signal-Prozess. Signal sei nun folgendermaßen definiert:

```
Signal = (green.red.up.down + green.up.red.down +
          green.up.down.red + up.down.green.red +
          up.green.down.red + up.green.red.down).Signal
```

Die anderen Prozesse bleiben unverändert. Überprüfen Sie mit `FSMCalc` wie in Aufgabe `25 c)`, ob ein Unfall auftreten kann oder nicht. Drucken Sie das Ergebnis von `FSMCalc` aus und geben Sie den Ausdruck mit ab. Interpretieren Sie schließlich das Ergebnis. Kann ein Unfall auftreten oder nicht? Begründen Sie Ihre Aussage.

**Aufgabe 27**

Verifizieren Sie das Modell des Bahnübergangs, wie es in der Einleitung definiert ist, mit `SPIN`. Gleich neben der Übungsangabe finden Sie das Skelett eines Bahnübergang-Modells in Promela: `crossing.pml`. Der Prozeß `Road` und der Checker-Prozeß `Checker` sind bereits enthalten. Vervollständigen Sie die Datei, indem Sie die Prozesse `Rail` und `Signal` hinzufügen. Rufen Sie dann `SPIN` mit der Option `-a` und der vervollständigten Datei auf (`spin -a crossing.pml`).

`SPIN` erzeugt einen Verifizierer in der Programmiersprache C (`pan.c`). Übersetzen Sie diese Datei mit ihrem C-Compiler und führen Sie das übersetzte Programm aus. Interpretieren Sie das Ergebnis. Kann es bei diesem Bahnübergang-Modell zu einem Unfall kommen? Begründen Sie ihre Antwort. Drucken Sie schließlich die vervollständigte Datei aus und geben Sie den Ausdruck mit einem Screenshot von der Ausgabe des von `SPIN` erzeugten Verifizierers ab.

**Aufgabe 28**

Wir modifizieren nun den Signal-Prozeß wie in Aufgabe `26`. Alle anderen Prozesse bleiben unverändert. Führen Sie eine Verifikation des Bahnübergang-Modells wie in Aufgabe `27` durch. Kann es bei diesem Bahnübergang-Modell zu einem Unfall kommen? Begründen Sie ihre Antwort. Drucken Sie schließlich die neue vervollständigte Datei aus und geben Sie den Ausdruck mit einem Screenshot von der Ausgabe des von `SPIN` erzeugten Verifizierers ab.