Übungen zur Vorlesung
Praktische Informatik II
Sommersemester 2001

Serie 4

Transitionssysteme

Aufgabe 1: Straßenbahnübergang (100%)

Die Einsatzmöglichkeiten von Transitionssystemen sind vielfältig. In dieser Aufgabe soll eine beispielhafte Verwendung zur Kontrolle eines Straßenbahnübergangs objektorientiert realisiert werden. Dieser ist in Abbildung 1 dargestellt.

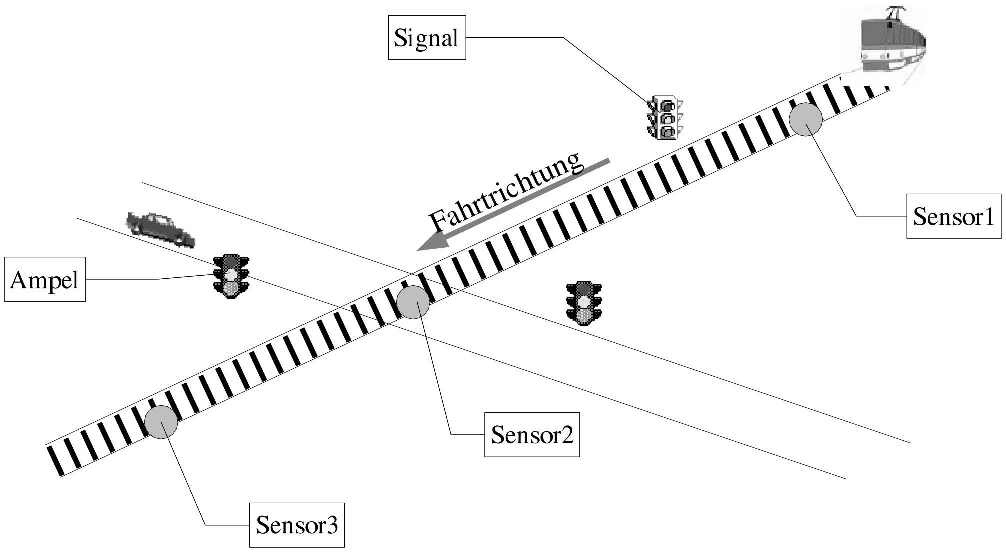

Abbildung 1: Straßenbahnkreuzung

Dabei handelt es sich um einen sehr vereinfachten Bahnübergang. Folgende wichtige Komponenten sind dort enthalten:

- **Ampel**: Sie kennt die Ausgaben **rot** und **grün** (gleichbedeutend mit **aus**). **grün** zeigt an, daß Verkehrteilnehmer den Übergang überqueren dürfen, **rot**, daß sie es nicht dürfen.
- **Signal**: Es kennt ebenfalls die Ausgaben **rot** und **grün**. Es zeigt an, ob die Straßenbahn fahren darf (**grün**) oder nicht (**rot**).
- **Sensor1**: Dieser Sensor zeigt an, daß sich eine Straßenbahn nähert, indem er die Eingabe **Sensor1** liefert.
- **Sensor2**: Dieser Sensor zeigt an, daß eine Straßenbahn gerade den Bahnübergang überquert, indem er die Eingabe **Sensor2** liefert.
- **Sensor3**: Dieser Sensor zeigt an, daß eine Straßenbahn gerade den Bahnübergang verläßt, indem er die Eingabe **Sensor3** liefert.

Die Straßenbahnlinie ist eingleisig, passierende Straßenbahnen fahren immer in der gleichen Richtung. Folgendermaßen soll der Bahnübergang funktionieren. Sobald eine Straßenbahn den ersten Sensor passiert, soll die Ampel auf **rot** und gleichzeitig das Signal auf **grün** geschaltet werden. Der zweite Sensor dient nur zur Kontrolle und zeigt dem System, daß die Straßenbahn tatsächlich den Bahnübergang überquert. Passiert die Straßenbahn nun den dritten Sensor, so wird die Ampel auf **grün** und das Signal auf **rot** geschaltet. Das System ist nun wieder bereit, die nächste Straßenbahn passieren zu lassen.

Daraus folgt, daß immer die Eingaben in der Reihenfolge **Sensor1**, **Sensor2**, **Sensor3** auftreten müssen. Insbesondere ist es nicht möglich, daß zwei nachfolgende Straßenbahnen in einem Zyklus durchfahren. In dem Fall, daß eine Verletzung der Eingabefolge auftritt, soll das System einen Fehlerzustand betreten, in dem sowohl die Ampel als auch das Signal auf **rot** stehen. In diesem Zustand verharrt das System, bis eine Technikerin das Problem behebt.

In Abbildung 2 ist ein Transitionssystem dargestellt, welches die oben beschriebene Funktionalität ausdrückt.

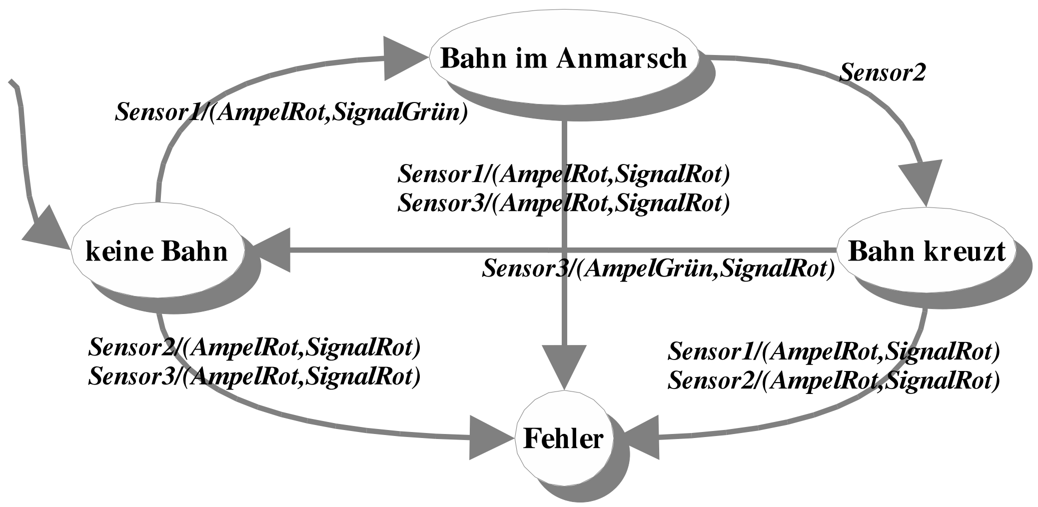

Abbildung 2: Transitionssystem für die Straßenbahnkreuzung. Mehrere Transitionen zwischen jeweils zwei Zuständen sind zu jeweils einem Pfeil zusammengefaßt.

Folgendermaßen soll die Realisierung dieses Transitionssystems geschehen: Entwerfen Sie eine abstrakte Klasse **TransitionSystem**, die folgende Methoden enthält:

public void createState (String stateName): Fügt einen Zustand mit dem Namen **stateName** in das Transitionssystem ein.

public void addTransition (String source, Label label, String target): Fügt eine Transition mit den angegebenen Quell- und Zielzuständen in das Transitionssystem ein. Das **label** enthält die Information, bei welcher Eingabe die Transition feuert und welche Ausgaben sie verursacht. Näheres zum Typ **Label** s. u.

public void traverseTransitionSystem(): Diese Methode aktiviert das Transitionssystem, d. h. eine Simulation (s. u.) des Systems wird ausgeführt.

Diese Klasse soll vollständig unabhängig von der konkreten Anwendung (Straßenbahnübergang) sein.

Insbesondere soll hier noch nicht festgelegt werden, auf welche Weise die Eingaben zum Transitionssystem gelangen. Deklarieren Sie dafür eine abstrakte Methode **protected Object getInput()**, die von abgeleiteten Klassen definiert wird und es so ermöglicht, konkrete Eingaben auf unterschiedliche Weise zu ermitteln.

Implementieren Sie die Klasse **TransitionSystem** so, daß sie selbst eine Liste von Zuständen enthält. Die Transitionen wiederum werden dann jeweils in ihrem Quellzustand verwaltet. Beachten Sie auch, daß einer der Zustände der Startzustand sein muß, damit die Simulation (s. u.) einen definierten Anfang hat. Sehen Sie dafür eine weitere Methode vor:

public void setInitialState (String stateName): Macht den Zustand mit dem Namen **stateName** zum Startzustand.

Beachten Sie bei der Implementierung der Methoden, daß möglicherweise ungültige Werte übergeben werden können, wie z. B. ein Quellzustand, der im Transitionssystem gar nicht existiert. Lassen Sie Ihre Methoden dabei angemessen reagieren. Beachten Sie vergleichbare Probleme auch in den anderen Klassen.

Entwerfen Sie eine Klasse **State**, die die Zustände des Transitionssystems repräsentiert. Insbesondere soll jedes Objekt dieser Klasse Referenzen auf genau die Transitionen enthalten, deren Quellzustand es ist. Ferner hat jeder Zustand einen Namen, der ihn identifiziert; folglich sollen sich zwei Zustände gleichen, wenn sie den gleichen Namen haben. Dafür ist es wie immer hilfreich, die automatisch von Object geerbte Methode **public boolean equals (Object o)** zu redefinieren.

Entwerfen Sie eine Klasse **Transition**, welche die Transitionen des Transitionssystems repräsentiert. Hier soll die Referenz auf den Zielzustand gespeichert werden. Weiterhin enthält sie Beschriftungen (Labels).

Entwerfen Sie dafür eine abstrakte Klasse oder Schnittstelle **Label**, welche den folgenden Zweck hat:

Jede Transition enthält ein Label. Im Label wiederum ist festgelegt, welche Eingabe die zugehörige Transition auslösen soll. Weiterhin ist im Label enthalten, welche Ausgaben erzeugt werden, wenn die zugehörige Transition ausgelöst wird. Damit nun sowohl **State** als auch **Transition** (wie auch **TransitionSystem**) anwendungsunabhängig bleiben, müssen Sie in **Label** zwei Methodendeklarationen vorsehen: Der Zweck der ersten ist, zu prüfen, ob ein gegebenes Ereignis mit dem im Label übereinstimmt. Die zweite wird benötigt, um die Ausgaben zu erzeugen. Dann können Sie **Label** verwenden, um es in **Transition** zu referenzieren.

Bis hierhin haben Sie eine allgemein verwendbare Klassenstruktur für Transitionssysteme geschaffen, in der die Simulation folgendermaßen ausgeführt werden kann:

1. In einer Endlosschleife wird jeweils eine Eingabe ermittelt.
2. Diese Eingabe wird dem aktuellen Zustand übergeben.
3. Der Zustand bietet allen seinen ausgehenden Transitionen diese Eingabe an, bis eine Transition durch diese Eingabe ausgelöst wird oder alle sie abgelehnt haben.
4. Jede Transition überläßt es dabei ihrem Label, zu entscheiden, ob sie ausgelöst wird. Falls ja, läßt sie das Label auch die Ausgaben erzeugen und gibt ihrem Quellzustand (welches der aktuelle Zustand ist) den Folgezustand zurück.
5. Der aktuelle Zustand gibt nun diesen Zustand an das Transitionssystem zurück. Wurde keine Transition ausgelöst, so gibt er sich selbst zurück.
6. Somit hat das Transitionssystem nun einen (neuen) aktuellen Zustand und setzt die Iteration fort.

Nun implementieren Sie konkrete Klassen, welche auf die Anwendung zugeschnitten sind. Das sind:

- Eine Klasse **TramController**, die von **TransitionSystem** abgeleitet ist. Sie soll das Ermitteln der Eingaben übernehmen. Eingaben sollen dabei die Zeichenketten "Sensor1", "Sensor2" und "Sensor3" sein. Sie sollen als interaktive Konsoleneingabe eingelesen werden (dafür können Sie die Klasse **Input** verwenden - benutzen Sie dafür die Version $Revision: 1.3 $). Andere Zeichenketten dürfen Sie ignorieren oder als Eingaben zulassen.

- Erstellen Sie zu jedem Label, welches Sie benötigen (also "Sensor1/(AmpelRot,SignalGrün)" etc.), eine Klasse, welche von **Label** abgeleitet ist bzw. implementiert. Implementieren Sie in der jeweiligen Klasse das Prüfen der entsprechenden Eingabe sowie das Erzeugen der jeweiligen Ausgaben. Dabei ist es hinreichend, die Ausgaben in Form von Textausgaben auf die Konsole zu realisieren.

Nun soll in der Hauptroutine des Programms ein **TramController** erstellt werden. Dieser wird dort zunächst mit Zuständen und Transitionen gefüllt und die Simulation gestartet:

```
TramController tramController = new TramController();
tramController.createState("keine Bahn");
tramController.createState("Bahn im Anmarsch");
tramController.createState("Bahn kreuzt");
tramController.createState("Fehler");
tramController.setInitialState("keine Bahn");
tramController.addTransition("keine Bahn",
                             new LabelSensor1AmpelRotSignalGruen,
                             "Bahn im Anmarsch");
...
tramController.traverseTransitionSystem();
```

Abgabe: 13. bis 15. Juni nach den jeweiligen Praktika. Die Abgabe soll sowohl elektronisch (Programm-Quellcode) als auch in gedruckter Form (mit LaTeX gesetzter kommentierter und erläuterter Quellcode) erfolgen!