Verkehrssimulation mit SystemSpecs
Traffic Simulation with SystemSpecs

Karl Guggisberg , Peter G. Kropf
Institut für Informatik und angewandt Mathematik
Universität Bern
Länggassstrasse 51
CH – 3012 Bern

22. Februar 1993

Zusammenfassung

In der vorliegenden Arbeit wird das Spezifikationswerkzeug *SystemSpecs* anhand der Beschreibung einer Verkehrssimulation und die Echtzeit-Implementation auf einem Transputernetzwerk dargestellt. Mit wachsender Verkehrsdichte in den Städten und dem Bedürfnis nach einem flüssigen, die Umwelt schonenden Verkehrsablauf, steigen die Anforderungen an Lichtsignalsteuerungen. Die implementierte Verkehrssimulation verfolgt wichtige Parameter einer Verkehrsanlage über die Zeit. Mit *SystemSpecs* wird ein modernes Werkzeug für den Entwurf von gefärbten Petri-Netzen (PRT-Netze) für die Implementation benutzt. Besonderes Gewicht wird auf einen modularen Netzentwurf gelegt. Durch *SystemSpecs* automatisch generierter occam-Code läuft auf einem Transputernetzwerk in Echtzeit ab.

CR Categories and Subject Descriptors: D.1.7, D.2.2, I.6.8, J.3
General Terms: Specification Tools, Petri Nets, Traffic Simulation, Parallel Computing

1 Einführung

Lichtsignalsteuerungen an Verkehrsknotenpunkten werden durch Programme gesteuert, für deren Entwurf verschiedene Optimierungsverfahren bekannt sind. Eine oder mehrere benachbarte Lichtsignalanlagen sollen zum Beispiel so gesteuert werden, dass die Kapazität eines Verkehrsnetzes maximiert wird. Dem Verkehrsingenieur stehen Programmpakete zur Verfügung, die optimale Steuerungen bezüglich verschiedener Kriterien entwerfen, wenn vorausgesetzte Parameter wie die Verkehrszusammensetzung oder die Verkehrsdichte bekannt sind(TRANSYT [11], SCOOT [8]).Oft fehlt dem Verkehrsingenieur jedoch ein Hilfsmittel, mit dem eine Lichtsignalsteuerung unter verschiedenen, leicht geänderten Rahmenbedingungen geprüft werden kann.

SystemSpecs ist ein modernes Petri-Netz Tool für gefärbte Petrinetze. In Zusammenarbeit mit dem Stadtplanungsamt Bern sowie der Firma TnTech[1], wurde in *SystemSpecs* ein Analyse- und Simulationstool für Verkehrsanlagen entworfen.

In den folgenden Abschnitten werden *SystemSpecs* und die damit realisierte Verkehrssimulation beschrieben.

[1] TnTech Parallel Computing Support AG, Waaghausgasse 2, 3001 CH – Bern

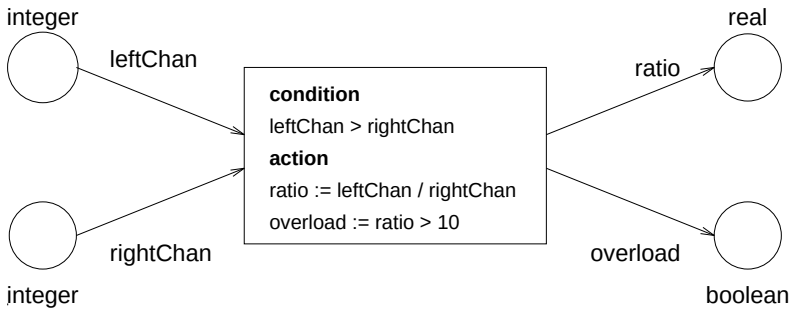

Abbildung 1: *SystemSpecs* Semantik

2 SystemSpecs

SystemSpecs[2, 1] ist ein Programm - Entwicklungswerkzeug, das sich besonders zur Spezifikation von komplexen Systemen eignet. Die formale Basis von *SystemSpecs* bilden dabei *Petri Netze höherer Ordnung*. Petri Netze sind Graphen, auf denen sogenannte *Marken* bewegt werden, und so die zeitliche Abfolge der Zustände des Systems beschreiben. Die mathematische Grundlage dieser Netze gestattet es, mit analytischen Methoden mögliche Probleme, z.B Verklemmungen (deadlocks), zu erkennen und gewünschte Eigenschaften nachzuweisen. Die grundlegenden Konstrukte eines parallelen Systems, wie Sie in **occam** etwa durch die `SEQ`, `PAR` und `ALT` Anweisungen gegeben sind, haben dabei einfache intuitive Darstellungen.

2.1 SystemSpecs Netze

Die von *SystemSpecs* verwendeten *Prädikat*–Transitionsnetze [3] eignen sich zur effizienten und sicheren Spezifikation komplexer paralleler Systeme. *SystemSpecs* Netze bestehen aus 4 Arten von Elementen: Marken, S-Elemente, T-Elemente und gerichtete Verbindungen zwischen S- und T-Elementen .

- *S-Elemente* sind Platzhalter für Daten (*Marken*) und werden mit Daten–Typen oder Konstanten beschriftet. Die Anwesenheit einer Marke auf einem S-Element, graphisch durch einen schwarzen Punkt dargestellt, zeigt das Vorhandensein gültiger Daten auf dem Element an.
- *T-Elemente* (Transitionen) erhalten als Beschriftung Bedingungen auf den Input–Daten, die erfüllt werden müssen, damit ein T-Element schalten kann (*Condition*–Teil). Beim Schalten werden alle Marken von den Input S-Elementen weggenommen und alle Output S-Elemente mit Marken belegt. Damit ein T-Element überhaupt schalten kann, müssen also alle Input S-Elemente mit einer Marke belegt sein und alle Output S-Elemente frei sein. Welche Daten die Output S-Element erhalten, ist den T-Elementen als weitere Beschriftung beigefügt (*Action*-Teil).
- S- und T-Elemente werden durch Pfeile verbunden, welche die Datenflussrichtung, das heisst die Bewegungsrichtung der Marken angeben, und als Beschriftung einen Variablenamen erhalten, auf den sich die Datenzuweisungen im Action–Teil beziehen können.

Der semantische Gehalt von *SystemSpecs* Netzen wird am einfachen Beispiel der Abbildung 1 klar.

Wichtig für die Spezifikation komplexer Systeme mit Hilfe von *SystemSpecs* ist die Eigenschaft, dass *SystemSpecs* Netze *hierarchisch* aufgebaut werden können. Dabei können einzelne S- oder T- Elemente zu ganzen *Subnetzen* ausgebaut werden (siehe Abbildung 2). Damit ist es möglich,

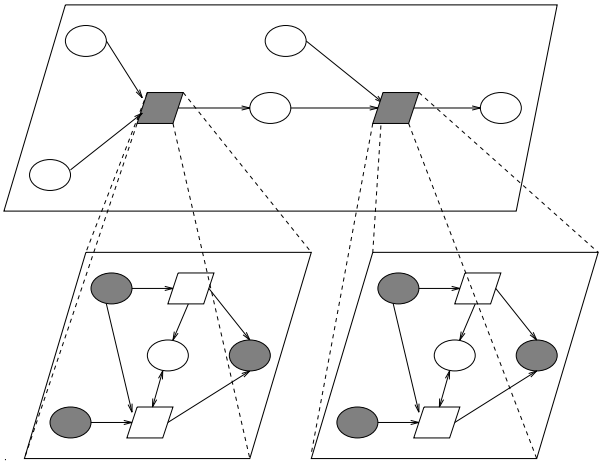

Abbildung 2: Netz–Hierarchie

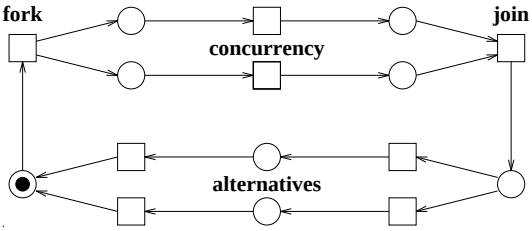

Abbildung 3: Parallelität und Nicht-Determinismus

den Systementwurf in verschiedenen Abstraktionsstufen in einem *top–down* Zugang zu realisieren. Zugleich garantiert diese Möglichkeit, dass auch umfangreiche Systeme kompakt und übersichtlich dargestellt werden können.

Mit *SystemSpecs* wird Parallelität graphisch ausgedrückt. Abbildung 3 illustriert die beiden Konstrukte, die den **occam** `PAR` und `ALT` Anweisungen entsprechen:

Nach dem Schalten der *fork*–Transition können die beiden folgenden Transitionen parallel ablaufen, während die *join* - Transition warten muss, bis beide beendet sind. Nach dem Schalten der *join*–Transition ist es vollkommen unbestimmt, welche der beiden nachfolgenden alternativen Transitionen schalten wird. Es kann jedoch nur eine der Alternativen schalten, da das Wegnehmen der einzigen Input–Marke die andere Alternative automatisch am Schalten hindert.

2.2 Graphische unterstützte Systemspezifikation und Animation

SystemSpecs erlaubt die hierarchische Spezifikation der Systeme, wobei das *SystemSpecs* Netz graphisch auf dem Schirm entworfen wird. Bedingungen und Transformationen (*Actions*), werden in der PASCAL–ähnlichen funktionalen Sprache *SpecsLingua* spezifiziert. Dieses Nebeneinander von Text- und Graphik erlaubt es, die verschiedenen Elemente des spezifizierten Systems mit dem jeweils geeigneten Mittel übersichtlich darzustellen: Während die Parallelität und die Prozess-Synchronisierung des spezifizierten Systems graphisch repräsentiert wird, können die sequentiellen Teile, d.h die Transformation der Daten, sowie die Bedingungen, unter denen die Transitionen schalten darf, mit textuellen Mitteln dargestellt werden.

Für die Analyse des spezifizierten Systemes kann das Netz oder ein ausgewähltes Subnetz mit einer graphischen Animation visualisiert werden. Der Benutzer kann den Fluss der Marken und

die durch die Marke übermittelten Daten auf dem Schirm verfolgen. Für die Ein- und Ausgabe von Daten stellt die *SystemSpecs* Benutzerschnittstelle graphische IO-Elemente zur Verfügung, die eine übersichtliche Darstellung von Modellgrössen in IO-Fenstern und die Dateneingabe über graphische Bedienungselemente gestattet.

3 SystemSpecs Netz Simulator

3.1 Das Programmiermodell

Ziel des parallelen Rechnens ist es, mit Hilfe von n Prozessoren eine Verkürzung der gesamten Rechenzeit um den Faktor n zu erreichen. Leider ist dies nicht immer einfach möglich: Einerseits kann der Programm-Algorithmus inhärent sequentiell sein, andererseits kann der Kommunikations–Overhead so gross werden, dass das Programm langsamer läuft als auf einem einzigen Prozessor. Die Aufgabe besteht aber nicht nur in der Entwicklung paralleler Algorithmen, sondern auch im Schaffen einer geeigneten Umgebung, in der die Programme effizient erstellt werden können.

Die Wahl des geeigneten Programmiermodells für die Implementierung eines *SystemSpecs* Netzes auf einem parallelen Rechner wird durch die folgende Liste von Eigenschaften, die *SystemSpecs* Netzen eigen sind, bestimmt:

Parallelität: T-Elemente ohne gemeinsame Input- und Output-Marken können unabhängig voneinander schalten.

Lokalität: Ob ein T-Element schalten kann, hängt nur gerade von seinen unmittelbaren Input- und Output-Marken ab.

Dezentralität: Konflikte, d.h. wenn T-Elemente die gleiche Input-Marke teilen, können ohne zentrale Instanz abgehandelt werden.

Nicht-Determinismus: Unbestimmtheiten treten in drei Punkten auf:

1. Es ist unbestimmt, wann genau eine Transition wirklich schaltet, wenn In- und Output-Marken es zulassen.
2. Im Falle eines Konfliktes ist unbestimmt, welche Transition schalten wird.
3. Welche der gültigen Kombination schliesslich zum Schalten einer Transition führt, ist nicht im Voraus bestimmt.

Mit der Theorie der kommunizierenden sequentiellen Prozesse (*Communicating Sequential Processes*, CSP) von C.A.R Hoare [4], steht ein Prozess–basiertes Programmiermodell zur Verfügung, das die Semantik von Petri Netzen wiedergibt.

In CSP kann ein *SystemSpecs* Netz als bestehend aus T-Prozessen und S-Prozessen betrachtet werden. Die T-Prozesse transformieren dabei Daten (Marken), die als Resourcen von den S-Elementen verwaltet weden. CSP steht auf einer fundierten mathematischen Basis, und kann als Liste von algebraischen Regeln formuliert werden. Wie ein *SystemSpecs* Netz kann in CSP ein System auf natürliche Weise in Teilsysteme geteilt werden, die parallel ablaufen können.

3.2 Der verteilte Netz–Simulator

SystemSpecs Netze und CSP kennen keine globalen Datenstrukturen und eignen sich daher als Grundlage für die Programmierung paralleler Systeme ohne globalen Speicher, deren Prozessoreinheiten über Kommunikationskanäle verbunden sind (*message passing interconnection architecture*).

Diese Lösung vermeidet den Engpass des Zugriffs auf einen globalen Speicher und ist daher besser geeignet, die grosse Anzahl, relativ einfacher Prozesse zu verwalten.

Für die Realisierung eines verteilten Netz-Simulators bietet sich der TRANSPUTER [5] an, dessen Mikroprogramm die Verwaltung vieler kommunizierender Prozesse – auch über die Prozessorgrenze hinweg – auf effiziente Weise unterstützt. Die nahe Verwandtschaft von *SystemSpecs* Netzen und CSP legt deshalb die Implementation der Netz-Simulation in **occam** – einer Sprache die in enger Anlehnung an die CSP Theorie entwickelt wurde – nahe. **occam** ist auf dem TRANSPUTER sehr effektiv implementiert, da viele **occam**-Kostrukte direkt im Instruktionssatz vorhanden sind.

SystemSpecs erzeugt automatisch eine auf Transputernetzwerken lauffähige **occam**-Simulation der spezifizierten Netze. Die Verteilung der einzelnen Prozesse, d.h der T- und S-Elemente, auf die zur Verfügung stehenden Prozessoren erfolgt nach dem in [6] beschriebenen Algorithmus ebenfalls automatisch.

4 Eine Verkehrssimulation als SystemSpecs Anwendung

Dieser Abschnitt beschreibt die Simulation einer Verkehrsanlage mit *SystemSpecs*.

4.1 Modellansätze

Modell für das zugrundeliegende Strassennetz

Ein Verkehrsknotenpunkt wird zerlegt in eine Menge von Netzsegmenten. Ein Netzsegment ist ein Strassenabschnitt, der von den Verkehrsteilnehmern frei befahren werden kann. Jedes Netzsegment ist durch eine Haltelinie begrenzt. Diese Haltelinie hat eine reale Entsprechung, wenn sie vor einem Lichtsignal liegt. Die Haltelinie wird nach [7] als virtuell bezeichnet, wenn sie keine diesbezügliche Entsprechung findet. Eine virtuelle Haltelinie kann zum Beispiel an der Stelle vorgesehen werden, wo die Ausfahrt einer Bushaltestelle in eine Fahrspur einmündet.

In Abbildung 4 ist dargestellt, wie zum Beispiel eine Verzweigung von Fahrspuren in Netzsegmente aufgeteilt wird.

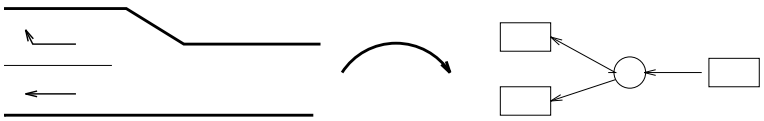

Abbildung 4: Beispiel für die Aufteilung in Netzsegmente

Modell für Verkehrsteilnehmer

In Petri-Netzen höherer Ordnung können die Marken komplexe Datentypen repräsentieren. Diese Eigenschaft wird ausgenutzt, um Verkehrsteilnehmer als attributierte Marken an den Einfahrquerschnitten in das Netz einzuspeisen und anschliessend durch das Netz zu führen.

Ein Verkehrsteilnehmer wird durch die folgenden Parameter beschrieben:

- Typ (PKW oder Lastwagen)
- Ziel (einer der möglichen Ausfahrquerschnitte)
- Wunschgeschwindigkeit (normalverteilt; Mittelwert und Standartabweichung wurden aus [7] übernommen)

- aktuelle Position relativ zum nächsten Haltequerschnitt
- aktuelle Geschwindigkeit
- aktuelle Beschleunigung

Unter den denkbaren Verkehrsteilnehmern, wurden für diese Simulation nur PKWs und Lastwagen berücksichtigt. Andere wichtige Verkehrsteilnehmer wie Busse und Strassenbahnen des öffentlichen Verkehrs, Fussgänger oder Radfahrer wurden vorerst ausgeklammert.

Modell für Lichtsignalanlagen

Lichtsignalanlagen werden nach verschiedenen Verfahren gesteuert. Kurzak unterscheidet in [9] Festzeitsteuerung, verkehrsabhängige Signalphasenwahl, verkehrsabhängige Grünzeitverteilung und verkehrsabhängige Signalplanbildung. In unserer Simualtion haben wird uns vorerst auf Lichtsignalanlagen mit Festzeitsteuerung beschränkt. Bei einer Festzeitsteuerung ist jedem Lichtsignal ein Signalplan zugeordnet, der den Signalzustand für jeden Zeitpunkt festlegt, und der mit einer gewissen Umlaufzeit zyklisch wiederholt wird. Es ist vorgesehen, in einem weiteren Ausbauschritt Sensoren zu simuliert und deren Daten für eine verkehrsabhängige Steuerung zu verwendet.

Modell für das Verkehrsverhalten

Eine wichtige Aufgabe der Simulation besteht darin, das Verhalten der Verkehrsteilnehmer möglichst realistisch nachzubilden. Verschiedene Ansätze sind denkbar. *Stochastische Modelle* gehen davon aus, dass genaue Messungen an den relevanten Querschnitten des Verkehrsknotens existieren. *Physikalische Modelle* beschreiben das Verkehrsverhalten mit einer Reihe von Differentialgleichungen. Wir haben uns für eine dritte Methode entschieden, die das Verkehrsverhalten *aufgrund von Regeln* festlegt. Unter der Annahme, dass Verkehrsteilnehmer nur in Abhängigkeit von ihrem direkten Vorgänger oder dem Zustand einer Lichtsignalanlage beschleunigen bzw. bremsen, haben wir eine Reihe von Regeln definiert, nach denen sich ein Verkehrsteilnehmer verhält. Diese Regeln basieren zum Teil auf aus der Fahrzeugfolgetheorie bekannten Abständen, wie dem *kritischen Abstand* oder dem *Folgeabstand* ([7]) und definieren Bedingungen, unter denen ein Fahrzeug beschleunigen darf bzw. bremsen muss. In diskreten Zeitschritten wird die aktuelle Position und das erwartete zukünftige Verhalten der Verkehrsteilnehmer gemäss diesen Regeln nachgeführt.

Die Vorteile dieses Ansatzes bestehen darin, dass das Fahrzeugverhalten jeweils nur lokal beeinflusst wird, womit eine Grundvoraussetzung für eine parallele Implementation gegeben ist. Als Nachteil müsste man anführen, dass diese Regeln unter Umständen nicht vollständig oder zu ungenau formuliert sind, um ein reales Verhalten der Verkehrsteilnehmer zu simulieren. Ein Testprogramm, das das Verhalten der Verkehrsteilnehmer visualisiert, hat uns jedoch in der Auffassung bestärkt, dass das Verhalten mit diesem Ansatz genügend genau an die Realität angenähert werden kann.

4.2 Implementation

In *SystemSpecs* wurden eine Reihe von elementaren Netzen erstellt, die miteinander kombiniert werden können, um eine Simulation aufzubauen.

Das wichtigste elementare Netz ist das *Netzsegment*. Dieses Netz ist in Abbildung 5 als Subnetz dargestellt. Die Plätze (`outSynch`, `tNS`, `fNS` etc) repräsentieren die Schnittstellen zu anderen Segmenten. Ein Netzsegment verwaltet Verkehrsteilnehmer in einer Warteschlange und führt deren Position in diskreten Zeitschritten nach. Ein weiteres elementares Netz generiert Verkehrsteilnehmer an den Einfahrquerschnitten und bestimmt die benötigten Initialparameter(Abbildung 6). Lichtsignale werden mit einem elementaren Netz gesteuert, das den Status des Lichtsignals nach

einer Festzeitsteuerung in Sekundenabständen neu festlegt. Kleinere elementare Netze wurden entworfen, um die Simulation zu kontrollieren, um eine Uhr nachzuführen, oder um den globalen Takt zu verlangsamen.

Mit Hilfe dieser elementaren Netze wurde eine Simulation für einen Verkehrsknotenpunkt in der Stadt Bern aufgebaut. Der Knoten besteht aus drei Teilknoten und umfasst sechs Einfahrquerschnitte, 24 Fahrspuren und 26 Lichtsignale. Die oberste Ebene des entworfenen Netzes ist in Abbidlung 7 dargestellt.

Der Verkehrsknoten, die Status der Lichtsignale und die Stauentwicklung werden graphisch visualisiert. Zu diesem Zweck wurden spezielle Ausgabeelemente für Lichtsignalanlagen und für die Staulänge entworfen. *SystemSpecs* bietet dafür einen Input/Ouput-Editor an.

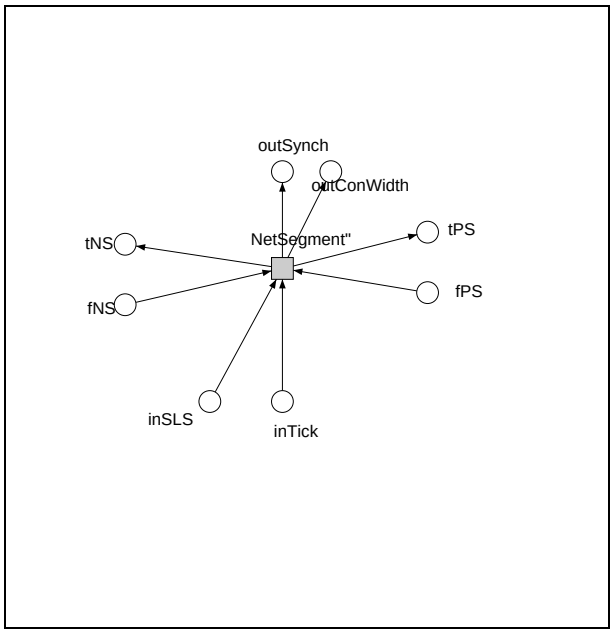

Abbildung 5: Elemetares Netz *Netzsegment*

5 Ergebnisse und weitere Arbeiten

Unsere Erfahrungen zeigen, dass sich *SystemSpecs* hervorragend eignet, um eine Verkehrsimulation aufzubauen. Weil man Netze in *SystemSpecs* hierarchisch gliedern kann, können auch sehr grosse Netze entworfen werden. Unsere Beispielsimulation umfasst ca. 1900 Plätze und Transitionen und ca. 3000 Pfeile, verteilt auf 6 Hierachieebenen.

Das Werkzeug *SystemSpecs* selbst ist in *Smalltalk* implementiert und auf verschiedenen Hardware Plattformen lauffähig (Sun, PC/Windows, Mac, etc.). Die direkte Simulation sehr grosser Netze innerhalb von *SystemSpecs* auf einer Workstation dauert im allgemeinen zu lange. Der durch *SystemSpecs* generierte occam - Code für die Verkehrssimulation konnte jedoch erfolgreich auf ein Tranputernetzwerk mit 8 Transputern abgebildet werden. Erst der Einsatz von Transputern ermöglichte eine Simulation in Echtzeit.

Die implementierte Verkehrssimulation ist in mancher Hinsicht ausbaufähig. Die gewählten Ansätze und Modelle sind mit Messungen zu validieren. Weitere elementare Netze für Bushaltestellen, einfädelnde Busspuren oder Kreisel sind zu entwerfen. Weiter könnte man die Ansätze verfeinern, indem man zum Beispiel jedem Verkehrsteilnehmer ein Mass für dessen Risikobereitschaft ([10])

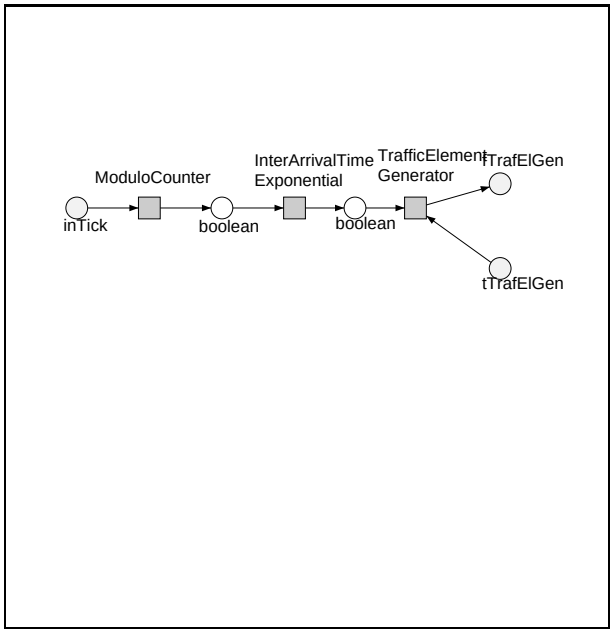

Abbildung 6: Generator für Verkehrsteilnehmer

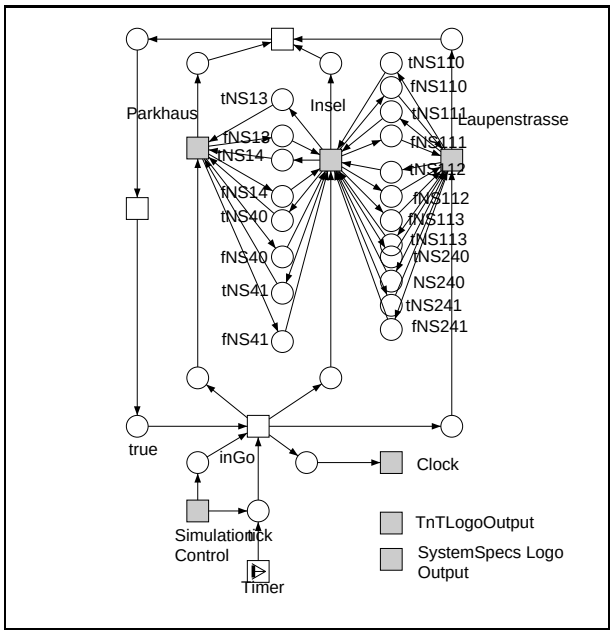

Abbildung 7: Oberste Ebene der Verkehrssimulation

zuordnen würde, oder indem man Überholvorgänge zuliesse. Zusätzliche wichtige Bewertungskriterien wie die durchschnittliche Anzahl Halte, die Verzögerungszeit oder die Reisezeit pro Fahrzeug könnten während der Simulation erfasst werden.

Literaturverzeichnis

[1] TnTech Parallel Computing Support AG. *SystemSpecs* Reference Manual. 1993.

[2] B. Bütler, R. Esser, and R. Mattmann. A Distributed Simulator for High Order Petri Nets. In *LNCS 483, Advances in Petri Nets 1990*, Berlin, 1991. Springer.

[3] W. Reisig. *Petrinetze, Eine Einführung.* LNCS. Springer, Berlin, second edition, 1986.

[4] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, Englewood Cliffs, 1985.

[5] INMOS. *Transputer Reference Manual.* Prentice-Hall, Englewood Cliffs, 1988.

[6] J.E. Boillat and P.G. Kropf. A fast distributed mapping algorithm. In H. Burkhart, editor, *CONPAR 90 – VAPP IV.* Springer, 1990.

[7] P. Philips, W. Stark, U. Weber *Untersuchungen von signaltechnischen Massnahmen zur Verbesserung des Busverkehrs in grünen Wellen.* Forschung Strassenbau und Strassenverkehrstechnik, Vol. 350, 1981

[8] P. B. Hunt et al. *SCOOT – a traffic responsive method of coordinating signals.* Transport and Road Research Laboratory Report 1014, 1981

[9] H. Kurzak *Untersuchungen zur verkehrsabhängigen Signalsteuerung von Stadtstrassenknotenpunkten mit Hilfe der Simulation.* Schriftenreihe des Institutes für Verkehrsplanung und Verkehrswesen der Technischen Universität München, Vol. 3, 1970

[10] W. Martin *Verkehrsablauf auf Stadtstrassen mit Lichtsignalanlagen.* Diss. Universität Karlsruhe, 1975

[11] M. Boltze *Optimierung von Umlaufzeiten in der Lichtsignalsteuerung für Strassennetze.* Diss. TU Darmstadt, 1988