Modellierung des Modellbahnmodells des Kicking Horse Passes durch gefärbte Petri-Netze Fortgeschrittenen Paktikum 1997

Lars Urbszat Wolfgang Hielscher

10. November 1997

Kapitel 1

Einleitung

Begleitend zur Vorlesung *Systemmodellierung mit Petri-Netzen* wurde ein Praktikum angeboten. Ziel dieses Praktikums war es, die in der Vorlesung vermittelten Grundlagen durch Erstellung eigener Petri-Netze zu vertiefen.
In der ersten Semesterhälfte wurden im Rahmen dieses Praktikums Systeme modelliert, die durch kleine Netze mit weniger als 30 Transitionen darstellbar waren. Die hierbei gesammelten Erfahrungen wurden in der zweiten Semesterhälfte genutzt, um ein komplexeres System zu modellieren. Die vorliegende Dokumentation beschreibt eben dieses komplexere System und seine Modellierung durch ein Petri-Netz.

1.1 Modelliertes System

Als zu modellierendes System wurde die (zur Zeit im Aufbau befindliche) Modellbahnanlage des Lehrstuhls für Rechnerorganisation gewählt. Auf dieser Anlage sollen 8 Züge weitgehend nebenläufig verkehren. Der Streckenverlauf der Anlage ist so gestaltet, daß u.a. der Betrieb des Kicking Horse Passes, eines eingleisigen Streckenstücks, das in beide Richtungen befahren wird, realisiert werden kann.
Die Modellierung sollte sich dabei nicht nur darauf beschränken, die Struktur der Anlage und die Position der Züge darauf wiederzugeben. Vielmehr sollte das Petri-Netz-Modell um Komponenten erweitert werden, die dafür sorgen, daß es im Petri-Netz nicht zu Belegungen kommt, die unerwünschten Sachverhalten auf der Anlage entsprechen, beispielsweise der Kollision zweier Züge. Die so gefundenen (Netz-)Mechanismen zur Steuerung von Zugbewegungen sollen dann als Grundlage für die Realisierung einer Steuerungssoftware für die Modellbahnanlage, im Rahmen einer Veranstaltung im folgenden Semester, dienen.

1.1.1 Der Kicking Horse Pass

In den westlichen kanadischen Rocky Mountains sind die Bahnstationen Lake Loise und Field durch eine eingleisige Bahnstrecke über den Kicking Horse Pass verbunden. Um die Berge mit geringer Steigung überwinden zu können, wurden zwei spiralenartige Tunnel angelegt. Der Tunnel im Cathedral Mountain ist 1,2 km lang und beschreibt einen Bogen von 291 Grad, der Tunnel im Mount Ogden ist 0,9 km lang und beschreibt einen Bogen von 217 Grad. Durch dieses Layout wurde erreicht, daß die Steigung auf der Strecke nicht mehr als 2,2% beträgt. Auf dieser Verbindung, die seit August 1909 in Betrieb ist, verkehren fast ausschließlich Güterzüge.

1.1.2 Die Modellbahnanlage

Der Kicking Horse Pass findet sich auch auf der Modellbahnanlage wieder: Auf einem eingleisigen Streckenstück werden über zwei Spiralen Höhenunterschiede überwunden. Im Gegensatz zur Realität enden aber die beiden Enden der Passstrecke in dem selben Bahnhof. So ist es möglich, die in der Wirklichkeit auftretende Situation, daß eine beliebig große Anzahl von Zügen den Pass überqueren soll, mit einer begrenzten Anzahl von Modellzügen nachzustellen, indem man einen oder mehrere Züge mehrfach den Pass durchfahren läßt.
Durch die Verbindung der Passenden in einem Bahnhof ergibt sich eine Ringstrecke, die im Folgenden Kicking Horse Pass genannt wird. Auf der Anlage finden sich noch zwei weitere Ringstrecken, der sogenannte Inner Circle und der Outer Circle, jeweils mit eigenem Bahnhof. Von jeder der drei Ringstrecken bestehen Verbindungen zu den jeweils anderen. Die drei Bahnhöfe dienen dazu, Züge abzustellen. Außerdem beginnen und enden Zugfahrten ausschließlich dort.
Eine detaillierte Beschreibung der Ringstrecken, ihrer Bahnhöfe und der Verbindungstrecken in Hinblick auf ihre Modellierung durch Petri-Netze folgt im Kapitel 2. Einen schematischen Übersichtsplan der gesamten Modellbahnanlage befindet sich im Anhang. Weitere Informationen über die Anlage sowie weitere schematische Pläne einzelner Streckenabschnitte findet man unter *http://www.informatik.uni-kiel.de/~cg/rail/html/gleisplan.html.*

1.1.3 Die Petri-Netz-Darstellung

Der Streckenverlauf der Modellbahnanlage ist in der Petri-Netz-Darstellung durch Stellen abgebildet, die in geeigneter Weise über Transitionen verbunden sind. Die Position der Züge auf der Anlage wird durch Belegung dieser Stellen mit entsprechenden Token wiedergegeben.
Durch weitere Netz-Strukturen werden die folgenden elementaren Anforderungen an eine Steuerung(ssoftware) modelliert:

- Die Kollision von Zügen muss ausgeschlossen werden, indem verhindert wird, daß mehr als ein Zug einen Streckenabschnitt belegt.

- Das Auftreten von deadlocks muss vermieden werden. Ein deadlock kommt dadurch zustande, daß die Fahrt zweier Züge gestoppt wird, da für eine kollisionsfreie Weiterfahrt das jeweils durch den anderen Zug belegte Streckenstück frei sein müßte.
- Die Wege, die die Züge auf der Anlage zurücklegen sollen, werden der Steuerung durch Fahrpläne genau vorgegeben. Die Steuerung darf von diesen Vorgaben nicht abweichen.

1.2 Mittel der Modellierung

In diesem Abschnitt wird ein kurzer Überblick über die Mittel der Modellierung, zum einen die verwendete Netzklasse und zum anderen die eingesetzte Software, gegeben. Da eine Vertrautheit mit Petri-Netzen vorausgesetzt wird, findet sich hier keine Einführung in gefärbte Petri-Netze, es werden lediglich die für unsere Modellierung der bahnanlage interessanten Eigenschaften dieser Netzklasse herausgestellt. Auch wird "Design/CPN", ein grafisches Modellierungswerkzueg für gefärbte Petri-Netze, nicht einführend beschrieben, sondern nur die für dieses Praktikum genutzten Programmfunktionen, Netzsynthese und schrittweise Darstellung der sich ändernden Tokenbelegung, erläutert.

1.2.1 Gefärbte Petri-Netze

Wie auch in einfacheren Netzklassen, z.B. S/T-Systemen, ist für das Schalten einer Transition in einem gefärbten Petri-Netz die Existenz einer ausreichenden Anzahl von Token auf den Stellen im Vorbereich dieser Transition notwendig. Da Stellen eine Farbe haben, sind die Token in der Regel nicht anonym, sondern durch ihre Beschriftung läßt sich genau spezifizieren, welche Objekte durch die Token dargestellt werden sollen. Durch die Färbung ist es möglich, verschiedene Stellen, die zur Modellierung eines bestimmten Sachverhaltes verwendet werden, beispielsweise eine Stelle und die dazugehörige Komplementärstelle, zu einer Stelle zusammenzufassen. Durch diese Faltung läßt sich das Petri-Netzmodell kompakter gestalten.
Neben dem Netzlayout und den Farben ist es in gefärbten Petri-Netzen möglich, mittels einer Inschriftensprache das Netzverhalten zu bestimmen. Die Inschriftensprache wird dabei verwendet für:

- **guards**: Das Schalten einer Transition kann an die Gültigkeit eines guards, der ein bool'scher Ausdruck der Inschriftensprache sein muß, gebunden werden. In diesem Ausdruck dürfen als Variablen die Bezeichner der Eingangskanten der entsprechenden Transition und globale Bezeichner verwendet werden. Somit spezifizieren guards die Faltung von Transitionen.
- **code action**: Beim Schalten einer Transition wird eine Folge von Anweisungen ausgeführt. Diese Anweisungen können beispielsweise Werte an

den Ausgangskanten dieser Transition berechnen oder dafür genutzt werden, das Schalten dieser Transition zu protokollieren. Ebenso wie guards beschreibt die code action die Faltung von Transitionen.

- **Ausdrücke als Kantenbeschriftung**: Diese Ausdrücke müssen die Farbe der Stelle haben, zu der die Kante führt bzw. von der sie ausgeht. Als Variablen dürfen, neben netzglobalen Variablen, die Bezeichner an den Kanten von und zu der Transition benutzt werden, mit der auch diese Kante verbunden ist.

Die Farben in Petri-Netzen entsprechen Datentypen in Programmiersprachen. Neben gängigen Wertemengen, wie z.B. Integer oder Char, lassen sich als Farben auch eigene Mengen definieren. Aus den einfachen Datentypen können ausserdem strukturierte Datentypen, wie Listen oder multi-sets, als Farben definiert werden.

1.2.2 Design CPN-Tool, CPN/ML

Das Design CPN-Tool unterstüzt den Entwurf und die Überprüfung von gegebenen Petri-Netzen. Als Inschriftensprache akzeptiert es dabei CPN/ML, welches eine spezielle Version der funktionalen Programmiersprache ML ist. Folgende drei Teile des Programms wurden im Rahmen diese Praktikums genutzt:

- **Graphische Editor**: In dem Editor lassen sich Petri-Netze in ihrer bekannte Darstellung, Transitionen als Rechtecke, Stellen als Kreise, Kanten als Pfeile, mit Hilfe der Maus erstellen. Hier können Stellen und Transitionen mit Labeln versehen werden, es werden hier die Kantenbeschriftungen, die Farben der Stellen und die guards und Code-Regionen der Transitionen eingetragen. Für eine Simulation des Netzes lassen sich ebenfalls hier die initialen Markierungen der Stellen festlegen.
  Damit bei größeren Projekten die graphische Darstellung nicht zu unübersichtlich wird, kann hier über ein transitionsbegrenztes Teil-Netz abstrahiert werden, indem es zu einer sogenannten Substitutionstransition zusammengefaßt wird. Diese kann dann wie eine normale Transition verwendet werden und somit ebenfalls Teil einer weiteren Substitutionstransition sein.

- **Syntax-Checker**: Durch diese Programmfunktion wird die syntaktische Korrektheit eines Petri-Netzes überprüft. Folgende Punkte werden dabei berücksichtigt:
  - alle Stellen müssen eine Farbe haben
  - die Bezeichner oder Konstanten an Kanten müssen die Farbe der Stelle haben, zu der sie führen bzw. von der sie kommen
  - die Initialmarkierungen der Stellen müssen die richtige Farbe haben
  - die Ausdrücke an Kanten oder in guards sowie die Code-Regionen werden auf korrekte ML-Syntax geprüft

- **Simulator**: Mit dem Simulator lassen sich mögliche Schaltfolgen des Petri-Netzes durchspielen. Dies kann entweder im Einzelschrittmodus erfolgen oder es wird solange fortgefahren, bis keine Transition mehr konzessioniert ist. Im letzteren Fall läßt sich die maximale Anzahl der Schritte beschränken, dies ist besonders für sich zyklisch verhaltende Systeme sinnvoll.
  In dem Simulator ist ein Schritt mit dem Schalten von genau einer Transition definiert. Das bedeutet für nebenläufig konzessionierte Transitionen, daß für die Durchführung eines Schrittes eine dieser Transitionen durch eine Zufallsfunktion ausgewählt wird, die dann schaltet. Ebenso verhält es sich, wenn im Laufe der Simulation mehrere Transitionen im Konflikt stehen, da auch dann zufällig eine Transition bestimmt wird, die dann schaltet.

Kapitel 2

Die Modellbahnanlage

2.1 Fahrtrichtungen eines Zuges

Auf dieser Modellbahnanlage ist festgelgt, daß eine Lokomotive ihre Waggons ziehen muß. Somit kann ein Zug nur in eine Richtung, vorwärts, fahren. Diese Einschränkung auf eine Fahrtrichtung gilt ebenso für Triebwagen.

2.2 Layout der Anlage

(Die grundlegende Struktur der Anlage ist bereits in Abschnitt 1.1.2 beschrieben.)
Aufgrund räumlicher Notwendigkeiten sind alle drei Ringstrecken konzentrisch aufgebaut, zu innerst der Kicking Horse Pass, dann der Inner Circle und als äußerstes der Outer Circle. Der Bahnhof des Kicking Horse Passes hat fünf Gleise, die Bahnhöfe der anderen beiden Ringstrecken jeweils drei.
Es ist, entsprechend dem Vorbild, vorgesehen, daß der Kicking Horse Pass in beide Richtungen befahren werden soll. Der Inner- und der Outer Circle soll hingegen wie eine Einbahnstraße genutzt werden, auf ihnen wird immer nur in eine Richtung gefahren, die sich nie ändert. Ebenso verhält es sich mit den Verbindungsstrecken zwischen den drei Gleisstrukturen.
Ein Zug, der in der Kicking Horse Pass Station steht, kann, bedingt durch die Führung der Verbindungsstrecken und die Einschränkung auf das vorwärts Fahren, direkt entweder nur den Outer Circle oder nur den Inner Circle erreichen, abhängig von seiner Ausrichtung im Bahnhof. Wenn er den Kicking Horse Pass zum Outer Circle verlassen hat und wieder in den Kicking Horse Pass zurückkehrt, wird dort nach der Rückkehr in genau der selben Richtung in der Kicking Horse Pass Station stehen wie vorher. Ebenso verhält es sich mit einem Zug, der vom Kicking Horse Pass zum Inner Circle gewechselt hat. Somit werden die beiden möglichen Fahrtrichtungen auf dem Kicking Horse Pass auf die gesamte Modellbahnanlage ausgedehnt.
Die einzige Möglichkeit, die Fahrtrichtung eines Zuges auf der Anlage zu ändern,

ihn also in bezug auf seine Ausrichtung in der Kicking Horse Pass Station zu wenden, bieten die Verbindungsstrecken zwischen dem Outer Circle und dem Inner Circle.

2.3 Blockstellen

Um organisatorisch eine Steuerung der Modellbahnanlage im Mehrzubetrieb realisieren zu können, ist der gesamte Streckenverlauf in einzelne Abschnitte, sogenannte Blockstellen, aufgeteilt. Jede dieser Blockstellen kann unabhängig von den anderen mit Fahrspannung versorgt werden. Um Informationen über die Positionen der Züge auf der Anlage zu haben, wurde jede Blockstelle mit Meßpunkten versehen.
Für unsere Modellierung im Petri-Netz sind wir idealerweise davon ausgegangen, daß direkt bestimmt werden kann, ob sich auf einer Blockstelle ein Zug befindet oder nicht und daß der Wechsel eines Zuges in die benachbarte Blockstelle keine Zeit kostet.
Als Blockstellen wurden zum einen Abzweigungen, Zusammenführungen und Kreuzungen und zum anderen Abschnitte nicht verzweigter Gleisstücke gewählt. Erstere werden im Folgenden als ,,verzweigte Blockstelle" referenziert, letztere als ,,nicht-" oder ,,unverzweigte Blockstelle". Um zu visualisieren, daß Züge durch die Steuerung auf nicht-verzweigten Blockstellen gestoppt werden können, sind diese mit Signalen ausgestattet. Ihre Stellung hat zwar keinen Einfluß auf die Fahrspannung des zugehörigen Gleisabschnitts, ein rotes Signal soll jedoch auf eine abgeschaltete Fahrspannung hinweisen. Bei der Einteilung der Blockstellen wurde berücksichtigt, daß ein Zug, der an einem roten Signal gestoppt wurde, sich in voller Länge auf dieser Blockstelle befindet.
Technisch ist es zwar möglich, die Fahrspannungen auf verzweigten Blockstellen abzuschalten, dies soll jedoch nicht für eine Steuerung genutzt werden. Folglich haben verzweigte Blockstellen keine Signale.
Jede Blockstelle trägt einen eindeutigen Namen. Der Name einer Blockstelle im Kicking Horse Pass beginnt mit KH, in Inner Circle mit IC und im Outer Circle mit OC. Ist die Blockstelle Teil des jeweiligen Bahnhofes, so hat sie das Infix ST, ist sie ein Kreuzung, dann JCT, SPL für eine Abzeigung, MRG für eine Zusammenführung und LN für ein unverzweigtes Gleisstück. Die Blockstellen der Bahnhöfe und der unverzweigten Gleisstücke tragen zudem eine fortlaufende Nummer als Namenssuffix. Ähnlich ist die Namenssystematik der Verbindungsstrecken.

Kapitel 3

Darstellung der Anlage im Petri-Netz

3.1 Darstellung von Blockstellen und Zügen

Jede Blockstelle der Modellbahnanlage ist durch genau eine Stelle im Petri-Netzmodell dargestellt. Auf der Modellbahnanlage kann sich auf den Blockstellen ein Zug befinden oder nicht, und diese Information muß in den Stellen im Petri-Netz wiederzufinden sein. Dies geschieht mittels der Färbung *BLOCK_INFO* dieser Stellen, die diese Unterscheidung ermöglicht:

color BLOCK_INFO = union Zug:ZUG_INFO + KeinZug

Das bedeutet, jede Stelle mit der Färbung *BLOCK_INFO* kann Token *KeinZug* oder Token der Farbe *ZUG_INFO* enthalten. Insbesondere wird durch das Netz gewährleistet, daß jede Stelle immer genau eines dieser Token enthält, da eine Blockstelle entweder durch genau einen Zug befahren wird oder sich kein Zug auf ihr befindet.

Ein Zug wird dargestellt als ein Tupel der Farbe *ZUG_INFO* und hat die Form:

*color ZUG_INFO = product STATE * ZUG_ID * DIR * SCHEDULE*

Bei der Beschreibung der Petri-Netzdarstellung der Modellbahnanlage wird statt von ,,einem Token der Farbe ZUG_INFO" kurz von ,,einem Zug" gesprochen. Das Tupel enthält die folgenden, einen Zug beschreibenden Informationen:

- **STATE**: Bei der Modellierung wird unterschieden zwischen fahrenden

und stehenden Zügen; ein Zug wechselt vom fahrenden in den stehenden Zustand, wenn ihm signalisiert wird, daß die folgende(n) Stelle(n) (noch) nicht befahrbar ist (sind) - und umgekehrt, ein Zug wechselt vom stehenden in den fahrenden Zustand, wenn ihm signalisiert wird, daß die folgende(n) Stelle(n) befahrbar ist (sind). Die Farbe *STATE* hat die Form *color STATE = with steht|faehrt.*
Der Inhalt dieser ersten Komponente des Zug-Tupels wird im Folgenden als „der Zustand des Zuges" bezeichnet werden.

- **ZUG_ID**: Zur Unterscheidung der einzelnen Züge werden sie indiziert, *ZUG_ID* ist eine Menge von Integer-Werten.

- **DIR**: Auf der Anlage gibt es zwei Fahrtrichtungen (s. Kap. 2.2), nenne sie nord und sued. Diese Bezeichnungen sind willkürlich und haben nichts mit den Himmelsrichtungen zu tun. Die Farbe *DIR* hat die Form *color DIR = with nord|sued.*

- **SCHED**: Die Züge auf der Anlage sollen Fahrplänen folgen (s. 1.1.3). Um die Bewegungen der Züge weitestgehend nebenläufig zu ermöglichen, wurden diese Fahrpläne in die Zug-Token durch die Farbe *SCHED* kodiert. *SCHED* ist eine Liste der Farbe *color DEST = with left|middle|right.*
Erreicht ein Zug eine Verzweigung, die für ihn keine Bahnhofseinfahrt darstellt, so gibt das Kopfelement der Liste an, in welcher Richtung diese überfahren wird. Das Auslesen der Information vernichtet das Listenelement. Die Einfahrt in einen Bahnhof erfordert hingegen keine Fahrplaninformation, da einem Zug ein beliebiges freies Gleis zugewiesen wird.
Da an jeder Verzweigung der Fahrplan eines Zuges verkürzt wird, liegt nach einer bestimmten Fahrstrecke ein leerer Fahrplan vor. In der vorliegenden Modellierung führt dies dazu, daß der Zug in einen speziellen Endzustand versetzt wird, der bis zum Ende der Simulation eine weitere Bewegung unterbindet. Anstatt den Zug in diese finale Parkstellung zu versetzen wäre es ebenso möglich, die ursprüngliche Fahrplaninformation dieses Zuges wiederherzustellen, um ihn somit endlos lange über die Anlage fahren zu lassen.

3.2 Unidirektional befahrene einfache Blockstellen

Siehe dazu Seite 8 im Anhang. Voraussetzung für eine Zugbewegung auf der Anlage von einer Blockstelle *LN* zu der in Fahrtrichtung folgenden *LN_<dir>* ist, daß der Zug auch wirklich faehrt und sich auf *LN_<dir>* kein Zug befindet. Im Petri-Netz muß dementsprechend das Zugtoken in *LN* den Zustand *faehrt* haben und in *LN_<dir>* muß sich das Token *KeinZug* befinden. Ist dies erfüllt, so ist die Transition *MOVE* aktiviert, und beim Schalten von *MOVE* werden die Token in *LN* und *LN_<dir>* vertauscht.
Befindet sich nun ein Zug auf Blockstelle *LN* und ist die nächste Blockstelle

LN_<dir> besetzt (durch einen anderen Zug), so muß der Zug anhalten. Dies wird auf der Anlage durch Signale dargestellt. Im Petri-Netz ist das Signal nicht explizit dargestellt, es ergibt sich implizit durch die *STATE*-Information der Züge: Ist in *LN* das Zugtoken mit dem Zustand *faehrt* und in *LN_<dir>* ebenfalls ein Zugtoken, so ist die Transition *STOP* aktiviert und beim Schalten von *STOP* wird der Zustand des Zugtokens in *LN* von *faehrt* zu *steht* modifiziert.
Wartet ein Zug auf Blockstelle *LN* vor einem Signal und ist die folgende Blockstelle *LN_<dir>* wieder frei, so schaltet das Signal den wartenden Zug frei. Im Petri-Netz: Ein Zugtoken befindet sich mit Zustand *steht* auf Stelle *LN* und in der folgende Stelle *LN_<dir>* ist das Token *KeinZug*. Dann ist die Transition *START* aktiviert, und beim Schalten von *START* wird der Zustand des Zugtokens in *LN* von *steht* wieder zu *faehrt* modifiziert.
Es ist klar, daß höchstens eine der drei Transitionen aktiviert sein kann, sie stehen nie in Konflikt zueinander. Zur übersichtlichen Gestaltung des Petri-Netzes wird dieses Subnetz in eine Substitutionstransition verwandelt, denn sie wird im Gesamtnetz öfters verwendet, es findet sich auf Seite 8 des Anhangs.

3.3 Signaldartstellung durch LookAhead-Stellen

Bei der obigen Darstellung von Zugbewegungen wird immer nur die folgende Stelle für Tokenverschiebung/Tokenmodifikation berücksichtigt. Das entspräche einem Signal auf der Anlage, das nur aufgrund des Zustandes der nächsten Blockstelle geschaltet wird. Nun gibt es aber Blockstellen auf der Anlage, auf denen Züge nicht halten dürfen, so daß gleich mehrere Blockstellen frei sein müssen, damit ein Zug durch ein Signal freigeschaltet wird. Demnach muß gewährleistet werden, daß diese Blockstellen frei sind, bevor ein Zug sie befährt. Im Petri-Netz wird dieses Problem durch sogenannte LookAhead-Stellen gelöst. Diese LookAhead-Stellen haben die Farbe *color LA_INFO = with frei|belegt* und enthalten Informationen, ob alle folgenden relevanten Blockstellen frei sind oder ob mindestens eine belegt ist. Diese Stellen enthalten immer genau eines der beiden Token *frei* oder *belegt*. Eine LookAhead-Logik steuert die Belegung der Stelle.
Befindet sich ein Zug auf Blockstelle *LN* und sind alle folgenden Blockstellen, auf denen kein Zug stehen darf und die erste, wo der Zug wieder halten darf, frei, so darf der Zug weiterfahren. Im Petri-Netz enthält die LookAhead-Stelle *LA_LN* der Farbe *LA_INFO* diese Information (siehe dazu Seite 5 im Anhang): Ist ein Zugtoken auf Blockstelle *LN*, so ist die Transition *MOVE* aktiviert, wenn *LA_LN* das Token *frei* enthält (die LookAhead-Logik schreibt ein *frei*-Token nach *LA_LN*, wenn alle diese folgenden Blockstellen das Token *KeinZug* enthalten). Schaltet *MOVE*, so wird das *frei*-Token konsumiert und ein *belegt*-Token nach *LA_LN* zurückgeschrieben.
Umgekehrt, befindet sich ein Zug auf Blockstelle *LN*, und ist eine der folgenden Blockstellen auf denen kein Zug stehen darf oder die erste, wo der Zug

wieder halten darf, durch einen anderen Zug belegt, so muß der Zug bereits auf *LN* anhalten. Im Petri-Netz enthält demnach die LookAhead-Stelle *LA_LN* ein *belegt*-Token. Dadurch ist die Transition *STOP* aktiviert. Schaltet *STOP*, so wechselt der Zustand des Zugtokens in *LN* von *faehrt* zu *steht* (analog zu 3.2). Das *belegt*-Token in *LA_LN* ist dabei nur Nebenbedingung und bleibt unverändert. Schaltet die LookAhead-Logik die Stelle *LA_LN* auf *frei*, so ist die Transition *START* aktiviert, das Zugtoken wird in den Zustand *faehrt* wechseln (auch hier ist das *frei*-Token in *LA_LN* nur Nebenbedingung) und kann mittels der Transition *MOVE* verschoben werden (siehe oben).
Auch dieses Subnetz wird mehrfach im Gesamtnetz verwendet und dementsprechend als Substitutionstransition realisiert. Die eben beschriebene Variante findet sich auf Seite 5 des Anhangs.

3.4 Nichtdeterministische Richtungswahl auf Splitweichen

Im Unterschied zu bidirektional befahrenen Strecken kommen bei den unidirektional befahrenen Strecken die Züge stets aus einer Richtung auf die Weichen. Deswegen kann man auf unidirektional befahrenen Strecken zwischen Splitweichen und Mergeweichen unterscheiden: Splitweichen spalten einen Streckenabschnitt zu mehrere Folgeabschnitte auf, Mergeweichen führen mehrere Streckenabschnitte zu einem zusammen. Die Schaltung dieser Weichen muß vor Ankunft eines Zuges feststehen. Im Petri-Netz wird zwischen Split- und Mergeweichen unterschieden, denn deren Darstellungen sind anders. Auch die Schaltung der Weichen geschieht nicht explizit, sondern ergibt sich aus den Fahrplänen *SCHED* der einzelnen Zugtoken. Im übrigen muß nur bei Splitweichen die Folgestrecke durch den Kopf der Liste *SCHED* bestimmt werden, denn bei Mergeweichen ist sie ja eindeutig.
In bestimmten Fällen ist die Weichenstellung von Splitweichen irrelevant für die Einhaltung von Fahrplänen. Dies betrifft die Einfahrt in die Bahnhöfe. Befindet sich ein Zug vor der Einfahrt in einen Bahnhof, so ist entscheidend, daß der Zug auf einem freien Haltegleis in den Bahnhof einfährt. Die Einfahrt in Bahnhöfe sollte also nicht Bestandteil des Fahrplanes sein, vielmehr sollte dem Zug ein beliebiges freies Haltegleis im Bahnhof bereitgestellt werden. Befindet sich beispielsweise ein Zug auf Blockstelle *OC_LN_6* des Outer Circle und soll über die Bahnhofweiche *OC_ST_0* - auf der der Zug nicht anhalten darf - in die Outer Circle Station einfahren, so muß gewährleistet sein, daß mindesten eines der drei Bahnhofgleise sowie auch die Weiche frei sind, ehe dem Zug die Einfahrt gewährt wird. So auch im Petri-Netz (siehe dazu Seite 25 im Anhang):
Befindet sich ein Zugtoken vor der Einfahrt in den Bahnhof auf Blockstelle *OC_LN_6*, so kann es nur in den Bahnhof bewegt werden, wenn die LookAhead-Logik in die LookAhead-Stelle *LA_OC_LN_6* ein *frei*-Token geschrieben hat (siehe 3.3). Das sollte geschehen, wenn *OC_ST_0* sowie eine der drei Bahnhof-

gleise ein *KeinZug*-Token enthält, und wird durch die Substitutionstransition (LookAhead-Logik) *LAL_OC_ST_0* geregelt (siehe dazu Seite 3 im Anhang). Ist ein *frei*-Token in *LA_OC_LN_6*, so wird das Zugtoken über *OC_ST_0* auf eines der Bahnhofgleise, auf dem *KeinZug* ist, bewegt. Gibt es davon mehrere, so wird nichtdeterministisch eines ausgewählt, die entsprechenden *MOVE*-Transitionen stehen im Konflikt bzgl. dem Zugtoken auf *OC_ST_0* (s. Seite 25).
Umgekehrt, befindet sich Zugtoken vor der Einfahrt in den Outer Circle Bahnhof auf Blockstelle *OC_LN_6* und ist entweder die Weiche *OC_ST_0* oder alle drei Bahnhofgleise besetzt durch andere Zugtoken, so muß der Zug angehalten werden.

3.5 Deterministische Richtungswahl auf Splitweichen (gemäß Fahrplan)

Im Gegensatz zu den Bahnhofeinfahrten gibt es auch Splitweichen, deren Stellungen relevant sind für die Einhaltung der Fahrpläne der Züge. So beispielsweise die Weiche *OC_SPL*: Hier kann der Zug entweder im Outer Circle bleiben oder in den Kicking Horse Pass wechseln. Der Fahrplan der Züge bestimmt die Stellung der Weichen. So auch im Petri-Netz: Das Zugtoken beinhaltet eine Fahrplanliste *SCHED*, die die Fahrstraße über die Weichen bestimmt. Der Listenkopf wird konsumiert und in eine neue Stelle *DEST_OC_LN_5* der Farbe DEST gegeben (siehe Seite 23 im Anhang). Je nachdem, ob der Kopf der Liste *SCHED* ein *left*-Token oder ein *right*-Token ist, ist die entsprechende *MOVE*-Transition in den jeweiligen Ctr aktiviert. Hier stehen nie zwei Transitionen im Konflikt.

3.6 Mergeweichen

Bei Mergeweichen werden mehrere Gleise auf eines geführt, d.h. mehrere Verbraucher greifen auf eine Resource zurück. Deswegen muß bei der Mergeweiche im Petri-Netz eine weitere Stelle für ein Resourcetoken verwendet werden, um einen wechselseitigen Ausschluß zu gewährleisten.
Betrachte die Blockstelle *OC_ST_4* auf der Modellanlage, auf die Züge von allen drei Bahnhofgleise des Outer Circle fahren können. Da sich nur ein Zug auf *OC_ST_4* befinden darf, muß, wenn mehrere Züge nach *OC_ST_4* fahren wollen, genau einem die Ausfahrt gewährt werden. Fordert nur ein Zug die Weiterfahrt nach *OC_ST_4*, so muß ihm ungehindert diese gewährt werden, und fordern mehrere Züge die Weiterfahrt nach *OC_ST_4*, so muß genau einem diese gewährt werden. Im Petri-Netz geschieht dies ebenfalls über LookAhead-Stellen und einer zugehörigen LookAhead-Logik. Die LookAhead-Stellen haben hier noch eine weiter Aufgabe: Sie geben an, welches Zugtoken den Zuschlag bekommen hat (siehe dazu Seite 26 im Anhang). Die LookAhead-Stellen *LA_OC_ST_1*, *LA_OC_ST_2* sowie *LA_OC_ST_3* sind initialisiert mit einem *belegt*-Token. Die Substitutionstransition *LAL_OC_ST_[123]*, deren Darstellung auf der Seite 4 des Anhang abgedruckt ist, steuert folgendermaßen die Ausfahrt:

Befindet sich genau ein Zug auf einem beliebigen Bahnhofgleis des Outer Circle, so wird in dessen LookAhead-Stelle ein *frei*-Token geschrieben (und ein *belegt*-Token konsumiert), wenn auf *OC_ST_4* ein *KeinZug*-Token befindet. Dabei wird das Resourcetoken aus *Res_OC_ST_4* konsumiert. Weitere sich im Bahnhof befindende Zugtoken können erst bewegt werden, wenn das Resourcetoken zurückgeschrieben wurde, wenn also das Zugtoken *OC_ST_4* verlassen hat. Mit anderen Worten: Dem Zug wird die Weiterfahrt gewährt, wenn kein Zug auf der Weiche *OC_ST_4* ist. Hat der Zug *OC_ST_4* verlassen, können weitere Züge den Bahnhof verlassen.
Befinden sich mehrere Züge auf mehreren beliebigen Bahnhofgleisen, so wird in genau einer LookAhead-Stelle ein *frei*-Token geschrieben (und ein *belegt*-Token konsumiert), wenn sich auf *OC_ST_4* ein *KeinZug*-Token befindet. Dieser Zuschlag erfolgt nichtdeterministisch, die einzelnen Transitionen, die den Zuschlag erteilen, stehen in Konflikt bezüglich des Resourcetokens. Es wurde dieses einfache Verfahren zur Konfliktlösung verwendet, da das vorliegende Modell weder Zeitvorgaben in den Fahrplänen berücksichtigen noch irgendwelche Züge vorrangig behandeln soll.
Hat ein Zugtoken den Zuschlag bekommen, so ändert es ggf. seinen Zustand von *steht* zu *faehrt*, und die Transition *MOVE* ist aktiviert. Schaltet *MOVE*, so wird - wie auch in 3.3 - das *frei*-Token in der LookAhead-Stelle konsumiert und ein *belegt*-Token zurückgeschrieben. Hat das Zugtoken dann auch *OC_ST_4* verlassen, und das Resourcetoken wurde zurückgeschrieben, so kann die LookAhead-Logik einen neuen Zuschlag erteilen.

3.7 Bidirektional befahrene einfache Blockstellen

Bei bidirektional befahrbaren Strecken ist auszuschließen, daß sich Züge entgegenkommen - die Einfahrt in bidirektional befahrbaren Streckenabschnitte wird in einer speziellen Logik gesteuert (siehe Kapitel 5). Aber hier geht es zunächst um die Darstellung von bidirektional befahrbaren Strecken im Petri-Netz.
Auch hier gilt: Genau eine Blockstelle der Modellanlage entspricht einer Stelle im Petri-Netz. Der Unterschied zu unidirektional befahrbaren Strecken nach 3.2 ist, daß für jede Stelle die *MOVE*, *START* und *STOP*-Transitionen für beide Fahrtrichtungen implementiert werden müssen. D.h. es gibt nunmehr die sechs Transitionen *MOVE north*, *START north* und *STOP north*, sowie *MOVE south*, *START south* und *STOP south* (siehe hierzu Seite 22 im Anhang). Je nachdem, wie die Fahrtrichtung im Zugtoken gesetzt ist, schalten die zugehörigen Transitionen. Auch hier ist klar, daß aufgrund der unterschiedlichen Kantenbeschriftungen höchstens eine der sechs Transitionen aktiviert sein kann, sie stehen nie in Konflikt zueinander.
Bei der Zusammenschaltung mehrerer bidirektional befahrbaren Blockstellen ist nunmehr sowohl die Belegung der nördlichen als auch die Belegung der südliche Blockstelle zu beachten. Ist beispielsweise ein Zugtoken im Kicking Horse Pass

auf *KH_LN_5* mit Fahrtrichtung *nord*, so ist die Belegung der Stelle *KH_LN_4* relevant für die Aktivierung der Transitionen *MOVE north*, *START north* und *STOP north*. Diese sind aktiviert analog zu den Transitionen der unidirektional befahrbaren Strecken. Für Zugtoken mit Fahrtrichtung *sued* ist entsprechend die Belegung der Stelle *KH_LN_6* relevant für die Aktivierung der Transitionen *MOVE south*, *START south* und *STOP south*.

Kapitel 4

Outer- und Inner Circle

Aus den genannten Grundstrukturen aus Kapitel 3 kann man leicht (bis auf wenige Ausnahmen) die unidirektional befahrenen Kreise Inner Circle sowie Outer Circle zusammensetzen. Anhand des Outer Circles seien im folgenden die einzelnen Komponenten des Kreises kurz erwähnt:

4.1 Outer Circle Station

Der Bahnhof des Outer Cirlcle wurde fast vollständig in Kapitel 3.4 und 3.6 beschrieben. Der Bahnhof ist eine Splitweiche, auf der nichtdeterministisch ein freies Bahnhofsgleis ausgewählt wird, in Verbindung mit einer Mergeweiche für die Ausfahrt aus dem Bahnhof. Berücksichtigt werden muß, daß das Resourcetoken der Mergeweiche beim Verlassen von *OC_ST_4* zurückgegeben werden muß. Ansonsten ist der Bahnhof durch zwei Dreierweichen einfach dargestellt.

4.2 OC_Split, OC_MRG und OC_JCT

Die Splitweiche *OC_SPL* ist ebenfalls in Kapitel 3.5 erklärt, die Mergeweiche entspricht der Beschreibung aus Kapitel 3.6. Die Junction *OC_JCT* wiederum ist nichts anderes als die Hintereinanderschaltung einer Zweier-Splitweiche mit einer Zweier-Mergeweiche. Auch hier müssen Resourcetoken zurückgegeben werden.

4.3 Inner Circle

Der Aufbau des Inner Circle ist ähnlich dem des Outer Circle. Die Fahrtrichtung auf dem Inner Circle ist sued, im Gegensatz zum Outer Circle. Der Bahnhof Inner Circle Station hat die gleiche Struktur wie die Outer Circle Station.
Einziger Unterschied: Es gibt anstatt einer Splitweiche, Mergeweiche und Zweier-Junction wie im Outer Circle hier eine Dreier-Junction.

Kapitel 5

Kicking Horse Pass

5.1 Der Kicking Horse Pass

Der Kicking Horse Pass ist ein Kreis, und dessen Start- und Endpunkt ist die Kicking Horse Pass Station. Der Pass selbst ist ein bidirektional befahrbarer Streckenabschnitt und als Petri-Netz einfach darzustellen; gemäß Kapitel 3.7 ist die Bewegung von Zugtoken innerhalb des Passes erklärt und daher uninteressant (zur Zusammenschaltung des Passes siehe Seite 21 im Anhang).

5.2 Die Kicking Horse Pass Station

Die Kicking Horse Pass Station (KHPS) ist das Herzstück der gesamten Anlage, denn hier findet nicht nur Einfahrtsregelung in den Pass statt, hier befinden sich neben den fünf Bahnhofgleisen auch die Anbindungen zu den anderen beiden Kreisen. Dementsprechend ist es notwendig, eine Vielzahl von Teilnetzen zu schaffen, um einen kollisionsfreien Betrieb des Bahnhofs zu erreichen. Diese zu Steuerung notwendigen Teilnetze sind dann, entsprechend den Steuerungsmechanismen des Inner und Outer Cicles, als Substitutionstransitionen zusammengefaßt. Das dadurch recht übersichtliche Petri-Netzmodell der KHPS ist auf Seite 29 des Anhangs abgebildet, die beiden Ein- und Ausfahrtsweichen *KH_ST_0* bzw. *KH_ST_9* finden sich auf den beiden folgenden Seiten des Anhangs. Der *Direction_and_Fairness_Controller*, der die Einfahrt auf den Pass kontrolliert, befindet sich auf Seite 40 und wird in Abschnitt 5.4 erläutert.
Die Kicking Horse Pass Station ist von zwei Seiten erreichbar, über die Junctions *KH_ST_0* sowie *KH_ST_9*. Das birgt eine Reihe von Probleme in bezug auf die Einfahrt auf die Bahnhofgleise. Es muß ausgeschlossen werden, daß zwei Züge, von beiden Seiten kommend, das gleiche Bahnhofgleis befahren. Hierfür werden Resourcetoken für jedes Bahnhofsgleis benötigt. Ferner muß für jedes Bahnhofgleis eine Stelle der Farbe *DEST* vorhanden sein, in der für einen ausfahrenden Zug dessen Bewegung über die Ausfahrtsweiche angegeben wird (siehe Kapitel 3.5). Schließlich kommen noch die LookAhead-Stellen für jedes der Bahnhofglei-

se hinzu, welche den Zuschlag für die Ausfahrt von Zügen anzeigen.
Das ergibt insgesamt schon zwanzig Stellen für die Darstellung der Bahnhofgleise selbst - wobei die Übersicht verloren ginge. Deswegen wurde hier nicht die Linie beibehalten, daß jede Blockstelle auf der Modellanlage einer Stelle im Petri-Netzmodell entspricht, sondern die fünf Bahnhofgleise auf eine einzelne Stelle gefaltet. Auch die Resourcestellen, Destinationstellen und LookAhead-Stellen wurden jeweils zu einer Stelle gefaltet, so daß nur noch vier Stellen zur Darstellung nötig sind (siehe Seite 29 im Anhang): Die Stellen *KH_ST_[12345]*, *Dest_KH_ST_[12345]*, *Res_KH_ST_[12345]* sowie *LA_KH_ST_[12345]*. Die Faltung der Stellen erfordert die Einführung der Farben
color KHS_LANE = int with 1..5,
*color BLOCK_INFO_5 = product KHS_LANE * BLOCK_INFO*,
*color LA_INFO_5 = product KHS_LANE * LA_INFO*,
*color DEST_5 = product KHS_LANE * DEST*,
die die Farben der gefalteten Stellen beschreiben. Jedes in der KHPS ankommende Zugtoken hat durch die Konsumierung eines Resourcetoken aus der Stelle *Res_KH_ST_[12345]* die Garantie auf ein freies Bahnhofgleis. Nichtdeterministisch wird ihm ein freies Bahnhofsgleis der Nummer *KHS_LANE* von 1 bis 5 zugewiesen. Als Tupel *KHS_LANE * ZUG_INFO* wird es nach *KH_ST_[12345]* geschrieben (wobei das Tupel *KHS_LANE * KeinZug* konsumiert wird). Analog wird in die Stelle *Dest_KH_ST_[12345]* das Kopfelement der Fahrplanliste *SCHED* als Tupel *KHS_LANE * DEST* abgelegt. Und auch der Zuschlag für ein sich aus der Station bewegendes Zugtoken erfolgt durch Austausch des Tupels *KHS_LANE * belegt* mit *KHS_LANE * frei* in der Stelle *LA_KH_ST_[12345]*, so daß immer eine eindeutige Zuordnung der jeweiligen Token mit den jeweiligen Bahnhofgleisen gegeben ist. Nur die Stelle *Res_KH_ST_[12345]* enthält anonyme Token, da sie nur die Garantie auf ein freies Bahnhofgleis geben, nicht aber ein bestimmtes zuweisen.
In der KHPS werden noch weitere Resource-Stellen benötigt, denn es bedarf für einen einfahrenden Zug nicht nur die Garantie auf ein freies Bahnhofgleis, sondern auch die Garantie der freien Zufahrt zum Bahnhofsgleis (bzw. der freien Ausfahrt). Die Anbindungsblockstellen *KH_ST_10* sowie *KH_ST_20* gehören ebenso zur KHPS wie die Weichen *KH_ST_0* und *KH_ST_9*. Und u.U. darf ein Zug auf keiner dieser Blockstellen anhalten (abhängig von der Ein- bzw. Ausfahrt). Also benötigen diese Abschnitte eine Resourcenverwaltung, um den wechselseitigen Ausschluß des Befahrens der Blockstellen zu gewährleisten (zur Funktionsweise dieser Resourcenverwaltung siehe Kapitel 5.3). Diese Resource-Stellen sind *Res_KH_ST_0*, *Res_KH_ST_9*, *Res_KH_ST_10* und *Res_KH_ST_20* - auch hier wäre eine weitere Faltung möglich.

5.3 Anbindung der Kicking Horse Pass Station an die restliche Anlage

Die Kicking Horse Pass Station ist über die Blockstellen *KH_ST_0* und *KH_ST_9* mit der restlichen Anlage verbunden. Diese beiden bidirektional befahrenen Junctions sind sich sehr ähnlich, so daß hier nur *KH_ST_0* beschrieben wird. Alle im folgenden erwähnten Transitionen und Stellen befinden sich, wenn nicht anders angegeben, auf der Seite 31 des Petri-Netzes.
Da *KH_ST_0* eine verzweigte Blockstelle ist, kann ein Zug auf ihr nicht angehalten werden. Somit ergeben sich die folgenden vier Möglichkeiten, wie diese Blockstelle überfahren werden kann:

5.3.1 vom Outer Circle zur Kicking Horse Pass Station

Ein Zug, der vom Outer Circle in die Kicking Horse Pass Station fahren soll, nimmt seinen Weg über *OC_SPL*, *KH_ST_10* und *KH_ST_0* in die Kicking Horse Pass Station. Da der Zug auf keiner dieser drei genannten Blockstellen halten kann bzw. soll, wird das Einfahren dieses Zuges nach *OC_SPL* über eine LookAheadLogic geregelt, die nur dann freie Fahrt gibt, wenn jeweils eine Resource für *KH_ST_10*, *KH_ST_0* und die Kicking Horse Pass Station verfügbar ist. Die Existenz eines Zug-Tokens auf *KH_ST_10* setzt somit voraus, daß diese Resource-Token verfügbar waren und im Augenblick nicht vorhanden sind.
Wenn also ein Zug-Token auf der Stelle *KH_ST_10* liegt, so kann dieses durch die Transition *Ctr_KH_ST_10* nach *KH_ST_0* weitergeschaltet werden. Dabei wird das Resource-Token für die Blockstelle *KH_ST_10* reproduziert (es wurde von der LookAheadLogik für *OC_SPL* verbraucht). Da sich nun auf *KH_ST_0* ein Zug-Token mit dem Wert nord der DIR-Komponente befindet, ist ausschließlich *MOVE_KH_ST_0_north* konzessioniert, die das Zug-Token in die Kicking Horse Pass Station hineinbewegt. Ein Schalten dieser Transition erzeugt dann wieder ein Resource-Token für *KH_ST_0*. Außerdem wird dabei das Kopfelement der Fahrplanliste abgespalten und in die Stelle *Dest_KH_ST_[12345]* geschrieben.

5.3.2 vom Kicking Horse Pass in die Station

Ein Zug, der den Kicking Horse Pass in nord-Richtung überquert hat, muß von *KH_LN_1* über *KH_ST_0* in die Kicking Horse Pass Station weitergeleitet werden. Dafür sind je ein Resource-Token für *KH_ST_0* und die Station notwendig. Wenn diese zwei Resource-Token vorhanden sind, so kann durch *LAL_KH_LN_1* in *LA_KH_LN_1* ein frei-Token produziert werden. Erst dann kann die Transition *Ctr_KH_LN_1_north* schalten, die das Zug-Token von *KH_LN_1* nach *KH_ST_0* bewegt. Wie in 5.3.1 wird dann unter Freigabe der Resource dieses Token in die Station weitergereicht.

5.3.3 aus der Kicking Horse Pass Station auf den Pass

Wie *MOVE_KH_ST_0_north*, so spaltet auch *MOVE_KH_ST_9_south* (s. Seite 30 des Petri-Netzes) bei einem Weiterschalten eines Zug-Tokens in die Kicking Horse Pass Station die oberste Fahrplaninformation nach *Dest_KH_ST_[12345]* ab. Somit gilt für eine Zug, der aus der Kicking Horse Pass Station über *KH_ST_0* auf den Kicking Horse Pass fahren soll, daß er in sued-Richtung fährt und das für ihn die Fahrplaninformation rechts vorliegt. Unter diesen Voraussetzungen kann *LAL_KH_ST_[12345]_south* (unter Berücksichtigung der erst im folgenden beschriebenen Einfahrtsregelung für den Kicking Horse Pass) prüfen, ob sowohl die Resource für *KH_ST_0* vorhanden ist, als sich auch auf *KH_LN_1* ein KeinZug-Token befindet. Ist beides gegeben, so kann diesem Zug durch ein frei in *LA_KH_ST_[12345]* die Möglichkeit zur Weiterfahrt signalisiert werden. Dieses Signalisieren verschiebt dann die Fahrplaninformation dieses Zuges aus *Dest_KH_ST_[12345]* nach *Dest_KH_ST_0*. Wenn dann ein Zug, dem eine freie Fahrt signalisiert wurde, durch *Ctr_KH_ST_[12345]_south* nach *KH_ST_0* geschaltet wird, so wird dabei ein Resource-Token nach *Res_KH_ST_[12345]* geschrieben. Da, wie oben erwähnt, sich in diesem Fall in *Dest_KH_ST_0* ein rechts befindet, ist durch das Zug-Token auf *KH_ST_0* nur *MOVE_KH_ST_0_south-_right* konzessioniert, das dann unmittelbar das Zug-Token nach *KH_LN_1* weiterschalten kann.

5.3.4 aus der Kicking Horse Pass Station zum Inner Circle

Der direkte Weg eines in sued-Richtung fahrenden Zuges aus der Kicking Horse Pass Station zum Inner-Cicle führt über *KH_ST_0* und *KH_ST_10* nach *KH_LN* (und von dort über *IC_JCT* auf den Inner Circle). Neben der Richtungskomponente sued gilt für einen derartigen Zug, daß für ihn in *Dest_KH_ST_[12345]* die Fahrplaninformation links vorliegt. Unter diesen Voraussetzungen signalisiert die LookAheadLogic LAL_KH_ST_[12345]_south nur dann freie Fahrt, wenn die Resource-Token für *KH_ST_0* und *KH_ST_10* vorhanden sind und sich auf der Stelle *KH_LN* ein KeinZug-Token befindet. (In diesem Fall ist die noch nicht beschriebene Einfahrtsregelung für den Kicking Horse Pass irrelevant.) Schaltet nun *LAL_KH_ST_[12345]_south* so wird auch in diesem Fall die Fahrplaninformation nach *Dest_KH_ST_0* weitergereicht. Dieses Schalten konzessioniert dann außerdem *Ctr_KH_ST_[12345]_south*, welches das Zug-Token aus dem Bahnhof nach *KH_ST_0* verschiebt und dabei ein Resource-Token für die Kicking Horse Pass Station produziert. Analog zum vorigen Fall kann dann durch *MOVE_KH_ST_0_south_left*, unter Erzeugung des Resource-Tokens in *Res_KH_ST_0*, das Zug-Token nach *KH_ST_10* durchgereicht werden. Der weitere Weg dieses Zuges nach *KH_LN* wird durch die Transition *MOVE* (auf Seite 21 des Petri-Netzes) bestimmt.

5.4 Einfahrtsregelung für den Kicking Horse Pass

Da die Pass-Strecke zum einen eingleisig ist und zum anderen in beide Richtungen befahren werden soll, ist es notwendig, die Einfahrt in den Pass zu reglementieren, um zu verhindern, daß sich Züge mit unterschiedlicher Fahrtrichtung auf dem Pass befinden. Weiterhin sind Mechanismen zu finden, damit die eine Fahrtrichtung gegenüber der anderen nicht benachteiligt wird.
Auf der Modellbahnanlage ist durch die Streckenführung bedingt, daß alle Züge, die in den Pass einfahren sollen, aus der Kicking Horse Pass Station kommen müssen. Somit nimmt die Einfahrtsregelung über *LAL_KH_ST_[12345]_south* (Seite 35) und *-_north* (Seite 33) auf die entprechenden Züge im Bahnhof Einfluß.
Im Petri-Netz ist die Einfahrtsregelung durch sechs Stellen und eine Substitutionstransition realisiert. Die Stellen haben dabei folgende Bedeutung:

- **DIRECTION_OF_NEXT_LEAVING_TRAIN**: Diese Stelle enthält immer genau ein Token der Farbe DIR. Es legt die Richtung fest, in welcher der nächste Zug in den Pass **einfahren** darf. Somit können sich beispielsweise noch in nord-Richtung bewegende Züge auf dem Pass befinden, wenn in dieser Stelle ein sued-Token befindet.

- **trains_on_KHP_south**, **trains_on_KHP_north**: In diesen beiden Stellen wird durch jeweils ein Token der Farbe INT gezählt, wieviel Züge sich in der entsprechenden Richtung im Augenblick auf dem Pass befinden. Es ist klar, daß dabei immer mindestens einer der beiden Zähler den Wert Null hat.

- **requests_KHP_south**, **requests_KHP_north**: Durch die INT-Token auf diesen beiden Stellen wird spezifiziert, wieviele Züge im Bahnhof aufgrund ihrer aktuellen, abgespalteten Fahrplaninformation nach Verlassen der Kicking Horse Pass Station in den Pass einfahren sollen.

- **Counter**: Der Wert des INT-Tokens auf dieser Stelle gibt an, wieviele Züge bereits in der Richtung, die in DIRECTION_OF_NEXT_LEAVING_TRAIN gespeichert ist, bereits in den Pass eingefahren sind.

- **LIMIT**: In dieser Stelle ist die maximale Anzahl von Zügen gespeichert, die vor einem Zug, der den Pass in nord-Richtung befahren soll, in suedrichtung in den Pass einfahren können.

Der Wert des Tokens auf der Stelle LIMIT wird nie geändert. Der Inhalt von DIRECTION_OF_NEXT_LEAVING_TRAIN wird ausschließlich durch den sogenannten KHP_Controller (Seite 40 des Petri-Netzes) geändert. Das Token auf Counter wird sowohl vom KHP_Controler als auf von den LookAheadLogiken fuer *KH_ST_0* und *KH_ST_9* modifiziert. Die Werte der Token auf den restlichen Stellen werden ausschließlich durch die beiden LookAheadLogiken geändert.

5.4.1 Einfluß der Einfahrtsregelung auf die LookAhead-Logiken

Die oben beschnriebene Einfahrtsregelung für den Kicking Horse Pass nimmt über die Modifikation der Token auf den Stellen DI-REC-TION-_OF-_NEXT-_LEA-VING-_TRAIN und Counter Einfuß auf die beiden LookAheadLogiken *LAL_KH_ST_[12345]_south* und -_north. Im folgenden wird das Verhalten des „suedlichen" LookAheadLogik (Seite 35 des Petri-Netzes) beschrieben:
Gesetzt den Fall, in *KH_ST_[12345]* befindet sich ein Zug-Token mit Richtungskomponente sued, dem noch nicht ein frei signalisiert wurde und für das die aktuelle Fahrplaninformation rechts vorliegt; außerdem sei die Stelle *KH_LN_1* mit einem KeinZug-Token belegt. Für ein Schalten von *LAL_right* muß als Nebenbedingung gelten, daß die aktuelle Fahrtrichtung auf dem Pass sued ist und daß sich keine Züge in nord-Richtung auf dem Pass befinden, dies ist durch die Doppelpfeile zwischen der Transition und den Stellen DIRECTION_OF_NEXT_LEAVING_TRAIN bzw. trains_on_KHP_north dargestellt. Weiterhin schränkt der guard an dieser Transition dessen Schalten auf zwei Fälle ein: entweder hat der Counter den Wert von LIMIT noch nicht überschritten oder der Wert von Counter hat das LIMIT überschritten, es befinden sich aber keine Züge in der Kicking Horse Pass Station, die in nord-Richtung über den Pass wollen. Der erste Fall sichert zu, daß eine bestimmte Anzahl von Zügen den Pass in sued-Richtung überqueren kann, auch wenn Anforderungen in der Gegenrichtung vorliegen. Im zweiten Fall hat die bestimmte Anzahl von Zügen den Pass bereits in diese Richtung durchfahren, da jedoch keine Anforderungen in Gegenrichtung vorliegen, wird auch weiteren Zügen in sued-Richtung die Einfahrt in den Pass ermöglicht. In diesem Fall wird ein in sued-Richtung den Pass überqueren sollender Zug nicht ohne Grund an der Einfahrt in den Pass gehindert.
Ein Schalten von *LAL_right* inkrementiert dann die Zähler für trains_on_KHP-_south und den Counter. Der Zähler für requests_KHP_south wird erst bei dem Schalten des Zug-Tokens nach *KH_ST_0* herabgesetzt.

5.4.2 Änderung der Einfahrtsrichtung auf den Pass

Der KHP_Controller (Seite 40 des Petri-Netzes) besteht aus 4 Transitionen, zwei für den Wechsel von nord nach sued und zwei für den Wechsel von sued nach nord. Es wird hier nur der Wechsel von sued nach nord beschrieben.
Die Transition Change S->N OverLimit schaltet, wenn mindestens LIMIT Züge in den Pass in sued-Richtung eingefahren sind und mindestens ein Zug in der Kicking Horse Pass Station steht, der in nord-Richtung über den Pass fahren soll. Durch diese Transition wird, in Verbindung mit den oben erwähnten LookAheadLogiken, sichergestellt, daß vor diesem in nord-Richtung über den Pass fahren sollenden Zug maximal LIMIT Züge in sued-Richtung in den Pass einfahren.
Die Transition Change S->N underLimit schaltet, wenn weniger als LIMIT Züge in den Pass in suedlicher Richtung eingefahren sind, außerdem keine weiteren

Anforderungen für den Pass in sued-Richtung vorliegen und Anforderungen für den Pass in nord-Richtung vorhanden sind. Dies folgt dem Ziel, daß ein Zug, der den Pass in nördliche Richtung durchfahren soll, nicht ohne Grund an der Einfahrt in den Pass gehindert wird.

Kapitel 6

Deadlocks und andere Probleme

6.1 Deadlock - die Gefahr einer Verklemmung

Bei einem Betrieb mit acht Zügen auf der Anlage bedarf es einer genaueren Kontrolle der Ein- und Ausfahrten, insbesondere in der Kicking Horse Pass Station. Es gibt immer noch Zug-Konstellationen, die die Lebendigkeit des Petri-Netzmodells zum erliegen bringen. So betrachte die folgende Situation: Es stehen fünf Züge mit Fahrtrichtung Nord in der KHPS, die in den Pass einfahren wollen. Einer bekommt den Zuschlag und fährt los. Inzwischen reserviert sich ein in Richtung Sued fahrender Zug , vom Inner Circle kommend - die Einfahrt in die KHPS, mit dem Ziel, in den Pass einzufahren. Dieser gelangt auch in die KHPS, und der deadlock ist perfekt..
Solch eine Situation ist nicht in dem vorliegende Petri-Netz berücksichtigt, und die Lösung dieses Problems bedarf einer aufwendigen externen Kontrolle, die eben diese Probleme verhindert. Diese müßte die Standortinformation aller Züge wissen sowie deren Fahrpläne, um evtl. rechtzeitig Züge anzuhalten. Das würde allerdings diesen Ansatz dieser Petri-Netzmodellierung aus dem Rahmen sprengen; eine einfache Lösung ist hier wohl nicht zu finden - die sich auch gerade mit der Gewährleistung von Fairness verträgt.

6.2 Andere Nachteile

Es mag sein, daß dieser Ansatz der Modellierung zu kompliziert ist im Hinblick auf den Einsatz von bestimmten Bausteinen. Sicherlich wurde eine ganze Reihe von Substitutionstransitionen erstellt, die sich im Netz öfters anwenden ließen, die Darstellung von Teilstrukturen der Modellanlage als Petri-Netz blieb aber häufig zu individuell, als daß auf Grundbausteine zurückgegriffen werden konnte. Das erklärt auch die hohe Anzahl an verschiedenen Netzdiagrammen.

Anhang A

Die Modellanlage dargestellt als Petri-Netz

Das folgende Inhaltsverzeichnis des Anhangs enthält die Seitenzahl und den Namen des jeweiligen Teilnetzes. Allerdings ist der Name des Teilnetzes häufig nicht identisch mit dem Namen der Substitutionstransition, durch die dieses Teilnetz in andere Netze integriert ist. Am offensichtlichsten wird dies bei den Substituionstransitionen für das Vorwärtsbewegen der Züge (z.B. Seite 5, 8, 12 oder 14). Die Namen dieser Teilnetze beginnen grundsätzlich mit ,,*BS_...*", die Substitiutionstransitionen tragen hingegen ,,*Ctr_...*" als Namenspräfix.
Die abgedruckten Netzpläne sind dafür gedacht, die geschilderten Teillösungen der Modellierung anschaulicher zu gestalten.
Zusätzlich zu den Netzplänen befindet sich ganz am Ende noch eine Seite mit dem Name ,,Hierarchy". Zum einen kann man dieser Darstellung die Beziehung der einzelene zueinander entnehmen und zum anderen findet man hier die Zuordung der Substitutionstansitionsnamen zu den entsprechenden Teilnetzen.

10. **BS_LA_ns_Rns**
11. **LAL_IC_JCT**
12. **BS_LA_ns_Dns**
13. nicht vorhanden
14. **BS_LA__Rns**
15. **Ctr_OCIC_LN_2**
16. **OuterCircle**
17. **Ctr_ICOC_LN_2**
18. **OC_MRG**
19. **LAL_OC_JCT**
20. **OC_JCT**
21. **KickingHorsePass**
22. **BS_LA__bi**
23. **OC_SPL**
24. **OC_Station**
25. **OCS_3_Split**
26. **OCS_3_Merge**
27. **BS_OC_Station**
28. **Prime**
29. **KH_Station**
30. **KH_ST_9**
31. **KH_ST_0**
32. **LAL_KH_ST_9_south**
33. **LAL_KH_ST_[12345]_north**
34. **LAL_KH_ST_0_north**
35. **LAL_KH_ST_[12345]_south**
36. **BS_KHS_Mrg**
37. **MOVE_KH_ST_[09]_<in_KHS>**

38. **BS_LA_ns_Rns**
39. **Ctr_KH_ST_9_north_right**
40. **KHP_Controller**