UNIVERSITÄT LEIPZIG

Beschränkte Petrinetze
Projektdokumentation

Autoren: Michael Große
Arne Brutschy

19. Februar 2003

University Leipzig
Postbox 100920
04009 Leipzig
Germany

Printed at Xylon Computersystems

Zusammenfassung

Dieses Dokument ist die vollständige Dokumentation zum Projekt „Beschränkte Petrinetze“. Im Rahmen des Projektes wurde eine Software entwickelt um bestimmte Eigenschaften selbiger zu simulieren und untersuchen. Das Dokument beschreibt die Planung und Implementierung der Software sowie die verwendeten Algorithmen.

Für Kommentare, Fragen oder Anregungen stehen wir gerne zur verfügung.

Leipzig, 19. Februar 2003

Michael Große[1]
Arne Brutschy[2]

[1] mgrosze@web.de
[2] abrutschy@xylon.de

Inhaltsverzeichnis

1 Aufgabenbeschreibung

1.1 Thema

Das Thema des Projektes ist es eine Software zu schreiben, die gewisse Eigenschaften von Petrinetzen betrachtet. Dabei sollen beschränkte Petrinetze mit Hilfe eines Erreichbarkeitsgraphen auf Lebendigkeit hin analysiert werden.

1.2 Bearbeiter

Die bearbeitenden Studenten sind Michael Große, Matr-Nr.8760302, und Arne Brutschy, Matr-Nr. 8964813.

1.3 Aufgabenstellung

Die Aufgabenstellung ist eine Software zu entwickeln, mit der man Petrinetze erstellen und simulieren kann. Zusätzlich sind verschiedene Algorithmen zu implementieren um bestimmte Eigenschaften der Netze zu überprüfen. Dabei soll hier die Beschränktheit gesichert werden und das Netz auf Lebendigkeit hin untersucht werden. Der Erreichbarkeitsgraph des Petrinetzes bildet dabei die Grundlage für die Analyse auf Lebendigkeit. Die interne Aufgabennummer ist die 5.b).

1.4 Ziele und Anwendungen

Die Ziele des Projektes sind:

- Erstellen eines beschänkten Petrinetzes
- Anzeige des Erreichbarkeitsgraphen
- Analyse auf Lebendigkeit

2 Inhaltsbeschreibung

2.1 Sachverhaltsbeschreibung

Petrinetze sind Modelle zur Beschreibung nebenläufiger Prozesse. Petrinetze können zahlreiche Eigenschaften tragen. In unserem Projekt ist aber nur die Beschränktheit von Interesse. Bei beschränkten Netzen, ist die Markenanzahl pro Stelle in jeder Markierung die von der Startmarkierung aus erreicht werden kann beschränkt. Ziel des Projektes ist es, ein beschränktes Petrinetz auf Lebendikeit zu überprüfen. Lebendigkeit bedeutet, daß alle Transitionen von einer, auf eine Anfangsmarkierung m_0 nach beliebig vielen Schritten folgenden, Markierung aus, geschalten werden können.

Beispiel:

Als Beispiel wollen wir eine Problemstellung aus der reales Welt nehmen. Es gibt einen Produzenten und einen Konsumenten, die den Austausch von Waren über ein Lager abwickeln. Das Lager hat nur eine bestimmte Kapazität, ist also beschränkt. Der Produzent kann eine Ware nur dann herstellen, wenn das Lager noch Kapazitäten frei hat. Auf der anderen Seite kann der Konsument wiederum nur Ware verbrauchen, solange sie im Lager vorrätig ist.

2.2 Modellierung

2.2.1 Petrinetz

In unserem Programm untersuchen wir beschränkte Petrinetze, d.h. wir müssen die Bestandteile von Petrinetzen in unserem Programm darstellen können. Dazu benötigen wir:

- **Petrinetz**
  Das Petrinetz $N = (S, T, F, K, W, m_0)$ besteht laut Definition aus Stellen, Transitionen, Kanten zwischen Stellen und Transitionen und umgekehrt, sowie Kapazitäten für die Stellen, Gewichten für die Kanten und schlußendlich einer Anfangsmarkierung.
  Ein Petrinetz an sich beschreibt nur den statischen Aspekt eines Petri-Systems. Der dynamische Aspekt (Schaltverhalten) kommt dann im nächsten Unterabschnitt zum tragen.
- **Stelle**
  Die Stelle nehmen in beschränkten Petrinetzen mehrere ununterscheidbare Marken auf. Dabei kann die Markenanzahl einer Stelle nur natürliche Zahlen als Were annehmen und nicht über die jeweilige Kapazität (Beschränkung) der Stelle steigen.

Abbildung 1: Grafische Darstellung einer Stelle

- **Transition**
  Für die Modellierung einer Transition ist neben der Verbindung mit Stellen vor allem die pre- und post-Menge, d.h. die Menge der Vorstellen bzw. der Nachstellen ausschlaggebend.

Abbildung 2: Grafische Darstellung einer Transitionen

- **Kante**
  Die Kanten sind durch die F-Relation des Petrinetzes beschrieben, d.h. sie legen die Beziehungen zwischen den beiden Typen Stelle und Transition fest. Eine Kante kann nur zwischen Stelle und Transition und Transition und Stelle existieren. Dadurch wird die statische Struktur des Netzes festgelegt. Mittels der W-Funktion ($W : F \rightarrow \mathbb{N}^+$) wird jeder Kante ein Gewicht zugeordnet. Das Gewicht gibt an, wieviel Marken eine Kante von einer Vorstelle beim Feuern einer Transition eingezogen werden oder nach dem Feuern in einer Nachstelle der Transition ablegt werden.

Abbildung 3: Grafische Darstellung einer Kante

- **Erreichbarkeitsgraph**
  Der Erreichbarkeitsgraph ist die graphische Darstellung aller möglichen Markierungen des Netzes. Jeder Knoten repräsentiert dabei eine Markierung. Wenn eine Transition existiert, die die Markierung $m^{'}$ nach $m^{''}$ überführt, werden die zugehörigen Knoten mit einer Kante verbunden. Diese wird mit der Transition beschriftet.
- **Lebendigkeit**
  Man bezeichnet ein Petrinetz als lebendig, wenn alle Transitionen des Netzes lebendig sind. Eine Transition t heißt lebendig, wenn sie unter allen Folgemarkierungen aktivierbar ist. Nicht aktivierbare Transitionen bezeichnet man als tot. Ein Petrinetz heißt tot, wenn alle enthaltenen Transitionen tot sind. Ein Petrinet das unter keiner Folgemarkierung tot tot ist, bezeichnet man als schwach lebendig.

Beispiel:

Wir wollen nun unser vorheriges Beispiel modellieren. Wir legen fest, daß das Lager eine Kapazität von drei haben soll. Außerdem soll der Konsument "schneller" als der Produzent sein, d.h. der Konsument kann in einem Schritt zwei Einheiten Ware verbrauchen, während der Produzent maximal eine Einheit auf einmal herstellen kann.

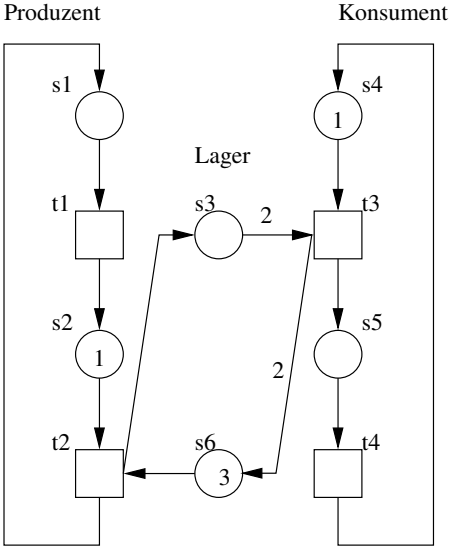

Abbildung 4: Beispielnetz

Das Lager wird durch die Stellen **s3** und **s6** dargestellt, wobei **s3** die Anzahl der durch Waren belegten Plätze und **s6** die Anzahl der freien Plätze im Lager angibt. Wie man hier deutlich sehen kann, wurde das Konsumentenverhalten mittels einem Kantengewicht von zwei vom Lager zum Konsumenten und umgekehrt modelliert. Der Anfangszustand ist des Netzes ist eine leeres Lager.

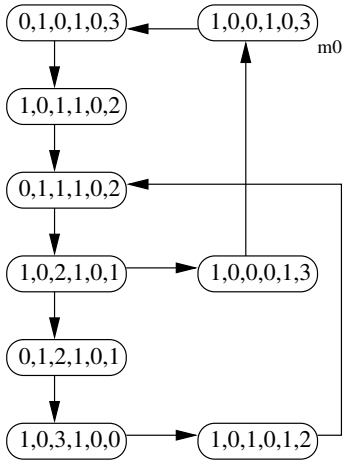

Abbildung 5: Erreichbarkeitsgraph des Beispielnetzes

2.2.2 Verhalten eines Petri-Systems

Das Verhalten eines Petri-Systems ist gekennzeichnet durch eine Anfangsmarkierung und eine Transitionsregel.

- **Markierung**
  Marken sind in Petrinetzen nicht unterscheidbar. Die Marken aller Stellen ergeben den Markierung (Zustand) eines Petrinetzes.
- **Transitionsregel**
  Eine Transitionsregel legt fest, wann eine Transition schaltet. Für beschränkte Petrinetze verwendet man die **sichere Transitionsregel**, d.h. eine Transaktion ist aktivierbar, wenn in den Vorstellen genügend Marken vorhanden sind und nach einem Schaltvorgang die Kapazität der Nachstellen nicht überschritten würde. Das ist genau dann der Fall, wenn das Kantengewicht der Kanten zur Transaktion kleiner oder gleich der Markenanzahl in der Vorstelle ist und die Markenanzahl in der Nachstelle zusammen mit dem Kantengewicht der Kanten nicht die Kapazität überschreiten.

2.3 Algorithmenbeschreibung

Zur Bestimmung der Lebendigkeit eines Netzes sind verschiedene Algorithmen notwendig. Diese sind fast vollständig aus dem Skript zur Vorlesung [1] entnommen.

- Inzidenzmatrix
  Für das Netz $N = (S, T, F, K, W, m_0)$, $s \in S$, $t \in T$ ist die Inzidenzmatrix

$$W_N = \begin{cases} t^+(s) = W(t,s) & \text{, falls} \quad s \in t^{\cdot} \setminus {}^{\cdot}t \\ -t^-(s) = W(s,t) & \qquad\quad\ \ s \in {}^{\cdot} t \setminus t^{\cdot} \\ \Delta t(s) & \qquad\quad\ \ s \in {}^{\cdot} t \cap t^{\cdot} \\ 0 & \qquad\quad\ \ sonst \end{cases}$$

- Erreichbarkeitsgraph bestimmen
- Lebendigkeitsanalyse

3 Softwarebeschreibung

3.1 Spezifikation

Die Software ist in zwei Teile unterteilt. Zum einen der Editiermodus, d.h. die Eingabe eines beschränkten Petrinetzen mit erstellen von Stellen, Transitionen, Kanten, Festlegung von Markenzahlen. Der andere Modus ermöglicht die Simulation des Schaltverhaltens des Netzes, d.h. die Möglichkeit feuerbare Transitionen zu feuern, die Anzeige der Inzidenz-Matrix, des Erreichbarkeitsgraphen und der Lebendigkeit des Netzes.

3.1.1 Globale Funktionen des Programms

Das Programm bietet mehrere Grundfunktionen:

- Laden und Speichern von Netzen im PNML-Format [2]
- Umschalten zwischen den Modi (Simulations- und Editiermodus)
- Anzeigen eines Hilfesystems

3.1.2 Eingabe von Netzen

Mit dem Programm ist es, wie oben beschrieben, möglich Petrinetze einzugeben.

- Einfügen von Transitionen (Festlegung eines Namens)
- Einfügen von Stellen (Festlegung eines Namens und der Kapazität)
- Einfügen von Kanten zwischen Stellen und Transitionen und umgekehrt
- nachträgliche Änderung aller oben genannter Parameter
- Änderung der initialen Markenanzahl von Stellen (unter Beachtung der Kapazität)
- Änderung der Kantengewichte (unter Beachtung der Kapazität der angrenzenden Stellen)
- Verschieben von Stellen und Transitionen inklusive automatischem Kantenlayout
- Löschen von Stellen und Transition mit Löschung angrenzender Kanten
- Löschen von Kanten
- Anzeige von Eigenschaften (Reinheit, Kontakfreiheit - evtl.)

3.1.3 Simulation

Das Programm ermöglicht es vom Editiermodus in den Simulationsmodus zu wechseln. Im Simulationsmodus kann das Netz bis auf die Positionen der Komponenten nicht verändert werden.

- Anzeige aktivierbarer Transitionen
- Anzeige aller Vor- und Nachbedingungen (evtl. wegen Übersichtlichkeit weglassen)
- Schalten aktivierbarer einzelner Transitionen
- Anzeige der Inzidenzmatrix (inklusive Markierung der feuerbaren Transitionen)
- Anzeige des Erreichbarkeitsgraphen
- Laden eines Zustandes durch Selektion eines Knotens im Erreichbarkeitsgraphen
- Analyse der Lebendigkeit des Netzes

3.1.4 Hilfe

Das Programm soll eine einfach Hilfefunktion bieten.

- Erklärung der Erstellung eines Petrinetzes
- Erklärung der Simulationsfunktionen
- Verweis auf die Webseite (Projektdokumentation, Literatur)

3.2 Implementierung

Die Implementierung wird plattformunabhängig mit Java erfolgen. Die Oberfläche wird mittels den Java Foundation Classes und Swing erstellt. Dies ermöglicht gleichbleibendes “look-and-feel“ auf allen Plattformen.

Das Laden und Speichern von Netzen soll mit Hilfe eines Standardformates geschehen um den reibungslosen Austausch von Netzen mit anderen Anwendungen zu ermöglichen. Ein Dateiformat, das diese Anforderungen erfüllt, ist in Form von PNML (Petrinet Markup Language) [2] schon vorhanden. Es basiert auf dem ebenfalls plattformunabhängigen XML.

3.2.1 Klassenstruktur

Abbildung 6: Übersicht der Klassenstruktur

Die Klassenstruktur gliedert sich in 4 Bereiche, die graphischen Komponenten, die Ereignisverarbeitung, das Modell des Petrinetzes und die Ansteuerung von Touchgraph. Die graphischen Komponenten stellen dabei die Daten aus dem Modell dar und werden bei Veränderung der Daten benachrichtigt. Die Ereignisverarbeitung manipuliert das Modell, z.B. Anpassung von Kantengewichten, Kapazitäten. Das Modell repräsentiert das Petrinetz und verwaltet dabei die Stellen, Transitionen und Kanten. Im Modell wird der Erreichbarkeitsgraph erstellt und die Lebendigkeit geprüft werden.

3.2.2 Touchgraph

Touchgraph ist ein frei verfügbares Graphendarstellungsframework, das nur einen Ausschnitt des Graphen darstellt und damit einen guten Überblick und bessere Performance bei der Darstellung des Erreichbarkeitsgraphen ermöglicht. Den Graphen kann man außerdem zoomen und verschieben.

3.2.3 Xerces

Xerces ist ein frei verfügbarer XML-Parser, der außerdem die Validierung von XML gegen ein XML-Schema ermöglicht. Dadurch können wir das Standardformat PNML (Petrinet Markup Language) in unserem Programm lesen und schreiben, was den einfachen Datenaustausch mit anderen Anwendung vereinfacht.

3.3 Anwendungsbeispiele

Hier werden die Ergebnisse des Programmes mit dem vorherigen Beispiel aufgeführt, sowie einige weitere Beispiele gegeben (totes/lebendiges Netz).

4 Ergebnisbeschreibung

4.1 Produkt, Grenzen und Erweiterungen

- das Programm kann nur beschränkte Petrinetze verarbeiten
- Laufzeit des Lebendigkeits-Algorithmus (Erfahrungserte folgen)
- es sind maximal 99 Marken pro Stelle möglich

Als Erweiterungen des Projektes kommen z.B. automatisches Layouten der Graphen, erstellen und bearbeiten von mehrseitigen/modularen Petrinetzen und weitergehende Überprüfung von Deadlocks in Frage.

4.2 Hard/Software-Voraussetzung

Java Runtime Environment 1.4 (Java Webstart empfohlen).
Weitere Hardwarevorraussetzungen folgen noch (Leistung).

A Anhang

A.1 Literatur

Literatur

[1] Prof. Dr. Siegmar Gerber: *Petrinetze - Skript*, Folien und Skript zur Vorlesung “Petrinetze“ an der Universität Leipzig, (2000), `http://www.informatik.uni-leipzig.de/theo/`

[2] Michael Weber und Ekkart Kindler: *The Petri Net Markup Language*, Humboldt-Universität zu Berlin, Technische Universität München, (2002), `http://www.informatik.hu-berlin.de/top/pnml/`

Index