Brandenburgische Technische Universität
Cottbus

Fakultät für Mathematik,
Naturwissenschaften und
Informatik

Ein Werkzeug zur Modellierung und Simulation von stochastischen Petrinetzen im Kontext biochemischer Netzwerke

Diplomarbeit

Eingereicht

von

Sebastian Lehrack

06.11.2007

Gutachter: Prof. Dr.-Ing. Monika Heiner
Prof. Dr. rer. nat. Wolfgang Marwan

Eidesstattliche Erklärung

Die Diplomarbeit wurde von mir selbständig angefertigt. Die verwendeten Hilfsmittel und Quellen sind im Literaturverzeichnis vollständig aufgeführt. Eingetragene Warenzeichen und Copyrights werden anerkannt, auch wenn sie nicht explizit gekennzeichnet sind.

Sebastian Lehrack

Danksagung

In erster Linie möchte ich meinen Eltern, Andrea und Bernd Lehrack, für ihre jahrelange Unterstützung, während meiner Studienzeit danken.

Einen besonderen Dank verdienen ebenfalls die folgenden Personen:

- Frau Prof. Heiner, die mich stets gefördert und gefordert hat,
- Herr Prof. Marwan und Frau Dr. Meyer für die Betreuung meiner Diplomarbeit und
- Ronny Richter, der mir praktische Hilfestellung bei der Ausarbeitung dieser Arbeit und beim Entwickeln des Werkzeuges gab.

Aufgabenstellung

In der vorliegenden Diplomarbeit sollen zum einen, die Grundlagen für die stochastische Modellierung von biochemischen Netzwerken beleuchtet und zum anderen, das Softwarepaket Snoopy um eine Netzklasse *Stochastic Petri Net* erweitert werden. Mittels dieser neuen Netzklasse sollen stochastische Petrinetze erstellt und simuliert werden können.

Für die Simulation ist eine angepasste Variante der direkten Gillespie-Methode zu entwerfen und zu implementieren. Die Ergebnisse der Simulation sollen in eine CSV-Datei exportiert oder graphisch als Plot dargestellt werden können.

Die Funktionsfähigkeit der programmierten Algorithmen soll, anhand ausgewählter biochemischer Beispielnetzwerke, nachgewiesen werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Auf dem Gebiet der Systembiologie etabliert sich immer mehr die Einsicht in die Notwendigkeit einer stochastischen Betrachtungsweise von biochemischen Netzwerken mit relativ kleinen Molekülanzahlen.

Mit dem Formalismus der stochastischen Petrinetze können die Erstellung und die Verständlichkeit von stochastischen Modellen wesentlich verbessert werden. Diese spezielle Petrinetzklasse dient dabei als Werkzeug zur Definition von stochastischen Prozessen, welche die eigentliche Semantik eines stochastischen Netzwerkes darstellen.

In dieser Arbeit werden die Modellierungsmöglichkeiten von stochastischen Petrinetzen, bezüglich biochemischer Reaktionsnetzwerke, untersucht, sowie das Softwarepaket Snoopy um eine entsprechende Netzklasse erweitert.

Die Diplomarbeit besitzt den folgenden Aufbau. Im zweiten Kapitel werden wir zwei spezielle Klassen von stochastischen Prozessen untersuchen. Als erstes wird dem Leser eine Einführung in die *diskrete Variante der Markow-Ketten* präsentiert. Aufbauend darauf werden *kontinuierlichen Markow-Ketten* untersucht, welche die semantische Grundlage der Basisklasse für stochastische Petrinetze bilden werden. Die Darstellung der mathematischen Grundlagen wird sich an Wilkinson [32] anlehnen.

Ziel des 3. Kapitel ist es, die Verbindung zwischen biochemischen Netzwerkmodellen, stochastischen Prozessen und stochastischen Petrinetzen zu knüpfen. Ein essentieller Punkt dieses Kapitels ist die Definition der Petrinetzklasse, die im Werkzeug Snoopy verwendet wird.

Der Darstellung von Simulationsalgorithmen für stochastische Petrinetze werden wir uns in Kapitel 4 widmen. Es sollen dort zwei typische Vertreter grundlegender Simulationsansätze vorgestellt werden. Unter anderem wird eine angepasste Version der *direkten Gillespie-Methode* präsentiert, welche im Werkzeug Snoopy implementiert wurde.

Kapitel 5 beinhaltet eine Beschreibung der Leistungsmerkmale, die der neu eingeführten Netzklasse in Snoopy zu eigen sind.

Spezielle Beispielanwendungen aus dem Bereich der biochemischen Netzwerke stehen im 6. Kapitel im Mittelpunkt. Wir werden unter anderem Steuerungsbausteine für den Zu- und Abfluss von Stoffmengen, bezüglich geschlossener Netzwerke, konstruieren.

Abschließend wollen wir in Kapitel 7 eine Zusammenfassung und einen Ausblick formulieren.

Bei der Präsentation der betrachteten Sachverhalte werden grundlegende Kenntnisse

der Wahrscheinlichkeitstheorie vorausgesetzt. Ebenfalls wird vom Leser erwartet, dass er mit der Netzklasse der diskreten Petrinetze vertraut ist. Für das Ausgleichen eventueller Defizite empfehlen wir jeweils [1] und [22].

Kapitel 2

Markow-Prozesse

2.1 Stochastische Prozesse

Unter einem stochastischen Prozess wird die mathematische Beschreibung eines zeitlich geordneten zufälligen Vorganges verstanden. Man ist damit in der Lage, Aussagen über den Zustand eines nicht-deterministischen Systems zu beliebigen Zeitpunkten zu treffen. Insbesondere können auch biochemische Systeme auf molekularer Ebene mit diesem fundierten Konzept beschrieben werden.

Ziel ist es dabei stets, die biochemischen Systeme so zu modellieren, das anhand durchgeführter Analysen am Modell, Rückschlüsse auf die realen Netzwerke gezogen werden können.

Charakteristisch für stochastische biochemische Systeme ist der Umstand, dass der Ausgang jedes Experimentes einen anderen messbaren Zeitverlauf der Stoffmengen aufzeigt. Alle theoretisch möglichen Verläufe bzw. Ausgänge des Experimente können in der *Menge der Elementarereignisse* Ω zusammen gefasst werden. Sie symbolisiert direkt die Ereignisse der realen Welt. In unserem Kontext wären das alle möglichen Verhaltensweisen eines vorliegenden biologischen Systems.

Um ein handhabbaren mathematischen Formalismus, in Form eines stochastischen Prozesses, zu erhalten, muss von der Repräsentationsmenge der realen Ereignisse Ω abstrahiert werden. Dazu wird zunächst eine eindeutige Beschreibung für den Zustand eines biochemischen Systems benötigt. Der durch uns favorisierte Ansatz benutzt für die Kodierung des Zustandes, die momentane Anzahl der Moleküle aller beteiligten Stoffe. Daraus folgt direkt, dass es sich bei der Menge aller möglichen Zustände um einen diskreten und abzählbaren *Zustandsraum* S handelt. Für ein wohldefiniertes und geschlossenes biochemisches System wird der Zustandsraum meist sehr groß, jedoch endlich sein. Dagegen kann für ein offenes System im Allgemeinen ein unendlicher Zustandsraum vorausgesetzt werden.

Die essentiellen Grundbausteine eines stochastischen Prozesses sind zeitlich geordnete Zufallsvariablen. Sie schlagen die Brücke von der Repräsentation der realen Experimentverläufe Ω zum mathematisch modellierten Zustandsraum S. Für einen simulierten Experimentverlauf im Modell nimmt jede Zufallsvariable $\chi_t : \Omega \rightarrow S$ einen Zustand an, in dem sich das modellierte System zum Zeitpunkt t befindet (Abb. (2.1)). Welche Zustände, zu welchen Zeitpunkten angenommen werden, ist zwar zufallsgesteuert, jedoch

nicht willkürlich. Die modellierte Struktur des Netzwerkes und die modellierten Wahrscheinlichkeitsverteilungen der Zufallsvariablen beeinflussen entscheidend die mathematische Realisierung eines künstlichen Experimentes.

Die Beziehungen zwischen biochemischen Netzwerken und stochastischen Prozessen sollen in den nächsten Kapiteln eingehend beleuchtet werden. Als Werkzeug für die Formulierung von stochastischen Prozessen werden wir im Laufe der Diplomarbeit den Formalismus der *stochastischen Petrinetze* vorschlagen.

Einleitend für die Darstellung der theoretischen Grundlagen, geben wir eine Definition, sowie eine Klassifikation für stochastische Prozesse an.

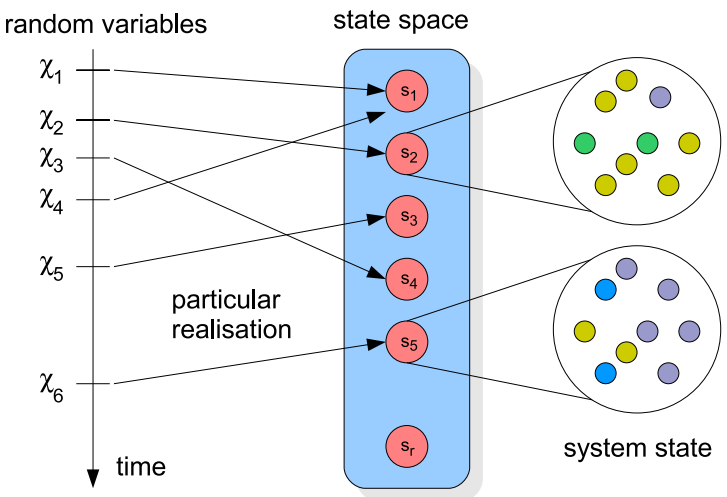

Abbildung 2.1: Ein diskreter stochastischer Prozess.

Definition 2.1 (Stochastischer Prozess)
Angenommen $(\Omega, \mathcal{F}, \mathrm{P})$ *ist ein Wahrscheinlichkeitsraum. Des weiteren, sei die Menge* $S \subseteq \mathbb{R}$ *der Zustandsraum des zu definierenden stochastischen Prozesses und die Menge* $T \in \{\mathbb{N}_0, \mathbb{R}_+\}$ *ein diskreter oder reeller Zeitindex. Ein stochastischer Prozess ist dann eine Familie von Zufallsvariablen*

$$\{\chi_t : \Omega \to S |\ t \in T\}\,.$$

Alle Zufallsvariablen sind somit über $(\Omega, \mathcal{F}, \mathrm{P})$ definiert, werden mit einem diskreten oder kontinuierlichen Zeitparameter $t \in T$ indiziert und bilden in den gemeinsamen Zustandsraum S ab.

Eine Klassifikation stochastischer Prozesse lässt sich zum einen, aus der Art des Zeitverlaufes und zum anderen, aus der Struktur des Zustandsraumes ableiten. Für beide orthogonalen Kriterien gibt es eine diskrete und eine kontinuierliche Ausprägung (Abb. (2.2)).

In dieser Arbeit werden wir uns im wesentlichen auf zwei Variationen, der so genannten *Markow-Prozesse*, konzentrieren. Die erste Variante besitzt einen diskreten, die zweite einen kontinuierlichen Zeitverlauf. Beiden gemeinsam ist der abzählbare diskrete Zustandsraum. Die beiden Klassen werden deshalb oft auch als *diskrete* bzw. *kontinuierliche Markow-Ketten* bezeichnet.

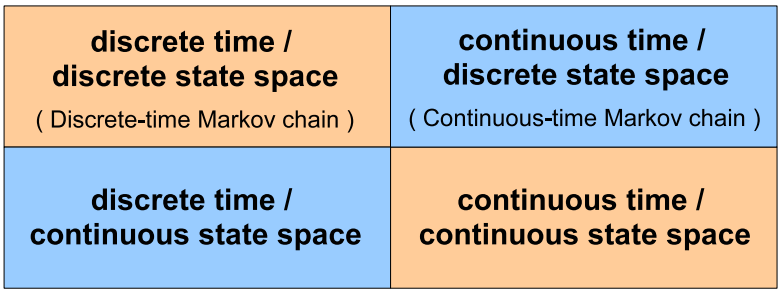

Abbildung 2.2: Verschiedene Klassen von stochastischen Prozessen.

2.2 Diskrete Markow-Kette

Eine diskrete Markow-Kette wird vor allem durch die *Markow-Eigenschaft* charakterisiert. Diese besagt, dass die zukünftige Entwicklung des modellierten Systems nur von der Gegenwart abhängt und nicht zusätzlich von der Vergangenheit. Befindet sich das betrachtete System in einem Zustand $s_i \in S$, ist es also für die Bestimmung der Wahrscheinlichkeiten für den weiteren Verlauf unerheblich, wie der Prozess in den Zustand s_i gelangt ist. Die Markow-Eigenschaft wird oft auch als Gedächtnislosigkeit bezeichnet und besitzt weitreichende Konsequenzen, auf die wir in späteren Abschnitten genauer eingehen werden.

Definition 2.2 (Diskrete Markow-Kette)
Angenommen $(\Omega, \mathcal{F}, \mathrm{P})$ *ist ein Wahrscheinlichkeitsraum. Der Zustandsraum* S *ist abzählbar und die Menge* $T \subseteq \mathbb{N}_0$ *spezifiziert einen Zeitindex. Die Menge von Zufallsvariablen*

$$\{\chi_t : \Omega \to S |\ t \in T\}$$

ist eine diskrete Markow-Kette, wenn die folgende diskrete Markow-Eigenschaft gilt

$$P\left(\chi_{n+1} = s \mid \chi_n = s_n,\ \ldots,\ \chi_0 = s_0\right) = P\left(\chi_{n+1} = s \mid \chi_n = s_n\right) \tag{2.1}$$

für alle $n \in \mathbb{N}_0$ *und* $s, s_0, s_1, \ldots, s_n \in S$ *mit* $P\left(\chi_n = s_n,\ \ldots,\ \chi_0 = s_0\right) > 0$.

Es soll im weiteren Verlauf der Arbeit das Akronym DTMC für diskrete Markow-Ketten benutzt werden, welches für die englische Bezeichnung *discrete-time Markov chain* steht.

Eine DTMC ist eindeutig beschrieben, wenn alle gemeinsamen Wahrscheinlichkeitsverteilungen, $P\left(\chi_n = s_n,\ \ldots,\ \chi_1 = s_1,\ \chi_0 = s_0\right)$ für $n \in \mathbb{N}_0$ und $s, s_0, s_1, \ldots, s_n \in S$, berechnet werden können. Dies ist im Allgemeinen nur schwer zu erreichen. Man behilft sich deshalb, mit der Betrachtung von Zustandsübergängen. Eine zentrale Frage lautet: Wie wahrscheinlich ist es, dass zum Zeitpunkt t, von einem Zustand s_i, in einen Zustand s_j gewechselt wird ?

Die Markow-Eigenschaft vereinfacht die Behandlung dieser Frage erheblich, da sie von der eigentlichen Vergangenheit abstrahiert. Sie gewährleistet, dass alle Information für den nächsten Zustandsübergang im aktuellen Zustand enthalten sind. Dies führt direkt zur *Einschritt-Übergangswahrscheinlichkeit*.

2.2.1 Übergangswahrscheinlichkeiten einer DTMC

Bevor die Definition der *Einschritt-Übergangswahrscheinlichkeit* angegeben wird, soll der Begriff der *Zeithomogenität* im Zusammenhang mit diskreten Markow-Ketten spezifiziert werden.

Definition 2.3 (Zeithomogenität)
Eine DTMC $\{\chi_t\}$ wird als zeithomogen bezeichnet, wenn gilt

$$P(\chi_{(m+n+1)} = s \mid \chi_{(m+n)} = s_n) = P(\chi_{(n+1)} = s \mid \chi_n = s_n) \tag{2.2}$$

für alle $m \in \mathbb{N}_0$.

Dies bedeutet, dass man Übergangswahrscheinlichkeiten unabhängig von diskreten Zeitverschiebungen betrachten kann. Wir wollen im weiteren alle Betrachtungen für zeithomogene Markow-Ketten durchführen.

Definition 2.4 (Einschritt-Übergangswahrscheinlichkeit einer DTMC)
Angenommen S ist der Zustandsraum einer DTMC $\{\chi_t\}$. Die Wahrscheinlichkeiten

$$P\left(s_x, s_y\right) := P\left(\chi_{n+1} = s_y \mid \chi_n = s_x\right), \qquad s_x, s_y \in S \tag{2.3}$$

werden als die Einschritt-Übergangswahrscheinlichkeiten von $\{\chi_t\}$ definiert.

Alle Einschritt-Übergangswahrscheinlichkeiten können in einer Matrix $\mathbf{P}$ zusammen gefasst werden.

Definition 2.5 (Übergangswahrscheinlichkeitsmatrix einer DTMC)
Angenommen $S = \{s_1, \ldots, s_r\}$ ist der diskrete Zustandsraum einer DTMC $\{\chi_t\}$. Die folgende quadratische Matrix wird als Übergangswahrscheinlichkeitsmatrix von $\{\chi_t\}$ definiert.

$$\mathbf{P} := \begin{pmatrix} P\left(s_1, s_1\right) & \ldots & P\left(s_1, s_r\right) \\ \vdots & \ddots & \vdots \\ P\left(s_r, s_1\right) & \ldots & P\left(s_r, s_r\right) \end{pmatrix}, \tag{2.4}$$

Die Matrix $\mathbf{P}$ ist eine stochastische Matrix.

Definition 2.6 (Stochastische Matrix)
Eine reelle $(r \times r)$-Matrix wird als stochastisch bezeichnet, wenn alle Elemente nichtnegativ sind und sich alle Zeilensummen zu eins ergeben.

Eine DTMC besitzt einen Graphen als einfache grafische Repräsentation. Im Graphen stellen die Knoten die verschiedenen Zustände des Systems dar. Die Kanten werden mit den jeweiligen Einschritt-Übergangswahrscheinlichkeiten beschriftet. Ist die Einschritt-Übergangswahrscheinlichkeit Null, wird die Kante weggelassen.

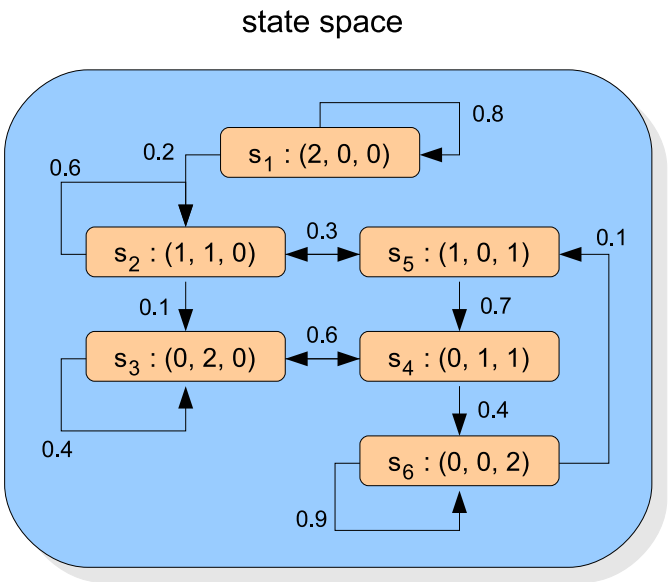

Abbildung 2.3: Die DTMC des Beispielnetzwerkes 1 als Graph.

Beispiel 1 (Netzwerk 1) *Wir betrachten ein einfaches biochemisches Netzwerk, bestehend aus den drei Stoffen A, B und C. Wenn wir als Ausgangsanzahl der Moleküle die Werte $A = 2$, $B = 0$ und $C = 0$ wählen, wollen wir sechs diskrete Zustände annehmen, siehe Abbildung (2.3). So kann zum Beispiel der Startzustand kodiert werden als $s_1 = (\#A, \#B, \#C) = (2, 0, 0)$. Wir setzen die Übergangswahrscheinlichkeitsmatrix für unser Beispielnetzwerk folgendermaßen an:*

$$\mathbf{P} = \begin{pmatrix} P(s_1, s_1) & \dots & P(s_1, s_6) \\ \vdots & \ddots & \vdots \\ P(s_6, s_1) & \dots & P(s_6, s_6) \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0.1 & 0 & 0.3 & 0 \\ 0 & 0 & 0.4 & 0.6 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0.4 \\ 0 & 0.3 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0.9 \end{pmatrix}$$

Das Quantifizieren dieser Übergangswahrscheinlichkeiten ist ein essentieller Teil für die Erstellung eines stochastischen Modelles. Es basiert meist auf dem aktuellen Verständnisstand und wird fortlaufend angepasst.

Die Einträge in der Matrix entsprechen den Kantengewichten des DTMC-Graphen (Abb. (2.3)), da es sich jeweils um eine Einschritt-Übergangswahrscheinlichkeit handelt. So ist zum Beispiel die Wahrscheinlichkeit, dass das System im nächsten diskreten Zeitschritt vom Zustand s_1 in den Zustand s_2 übergeht: $P(s_1, s_2) = 0.2$. Dies bedeutet, wenn zwei Moleküle vom Stoff A vorhanden sind und keine Moleküle für die Stoffe B und C vorliegen, ist die Wahrscheinlichkeit, das ein Molekül von A in ein Molekül von B übergeht gleich 20%. Die numerische Angabe dieser Wahrscheinlichkeit befindet sich in der Matrix an der Position (1, 2). Die Wahrscheinlichkeit für den Verbleib im Zustand s_1 im

nächsten diskreten Zeitschritt beträgt $P(s_1, s_1)$ =0.8 (Matrixposition (1,1)).

Mit der Übergangswahrscheinlichkeitsmatrix $\boldsymbol{P}$ und einer gegebenen Startkonfiguration ist das stochastische Verhalten des Systems vollständig beschrieben. Wir sind damit in der Lage Wahrscheinlichkeiten für konkrete Simulationsläufe zu berechnen.

Es wird hier beispielhaft die Wahrscheinlichkeit für einen Simulationsverlauf ermittelt, der die Zustandsfolge

$$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_5 \rightarrow s_4 \rightarrow s_6$$

besitzt. Daraus folgt, dass die Realisierungen der ersten sechs Zufallsvariablen des stochastischen Prozesses durch

$$run_1 \;:=\; (\chi_0 = s_1, \chi_1 = s_1, \chi_2 = s_2, \chi_3 = s_5, \chi_4 = s_4, \chi_5 = s_6).$$

bestimmt sind. Setzt man die modellierten Einschritt-Übergangswahrscheinlichkeiten an, ergibt sich für die Gesamtwahrscheinlichkeit

$$\begin{aligned} P(run_1) &= P(s_1,\ s_1) \cdot P(s_1,\ s_2) \cdot P(s_2,\ s_5) \cdot P(s_5,\ s_4) \cdot P(s_4,\ s_6) \\ &= 0.8 \cdot 0.2 \cdot 0.3 \cdot 0.7 \cdot 0.4 \\ &= 0.01344. \end{aligned}$$

2.2.2 Stationäre Zustandsverteilung

Eine der wichtigsten Fragestellungen bei der Betrachtung eines biochemischen Systems ist die, nach dem Vorhandensein eines so genannten *Fließgleichgewichtes* (Englisch: steady-state). Das Fließgleichgewicht beschreibt den Umstand, dass für jeden beteiligten Stoff die Verbrauchs- sowie die Produktionsmengen in den Reaktionen des Systems ungefähr gleich sind. Das System befindet sich demnach in einem *stationären Zustand*. Dass heißt, die Stoffmengen verändern sich, von außen betrachtet, nicht mehr. Die Stationarität bezieht sich jedoch nur auf die Beobachtung der Stoffmengen, die Reaktionen finden weiter fortlaufend statt. Für die Markow-Ketten entspricht der stationäre Zustand des Fließgewichtes einer *stationären Zustandsverteilung*. Aufbauend auf unserer bisherigen Terminologie, wollen wir uns diesem Begriff schrittweise nähern.

Zunächst sollen die Wahrscheinlichkeiten $P(\chi_n = s_i)$ für ein $s_i \in S$ betrachtet werden. Sie geben jeweils an, wie hoch die Wahrscheinlichkeit ist, dass sich das System nach n Zeitschritten im Zustand s_i befindet. Für diese speziellen Wahrscheinlichkeiten wird die Notation $\pi^n_{s_i}$ eingeführt. Erweitert man diese Notation zu einem Vektor, erhält man die *Zustandsverteilung* π^n nach n Zeitschritten.

Definition 2.7 (Zustandsverteilung nach n Zeitschritten für eine DTMC)
Angenommen $S = \{s_1, s_2, \ldots, s_r\}$ ist der diskrete Zustandsraum einer DTMC $\{\chi_t\}$. Wir definieren

$$\pi^n := (\pi^n_{s_1}, \pi^n_{s_2}, \ldots, \pi^n_{s_r}) = (P(\chi_n = s_1),\ P(\chi_n = s_2),\ \ldots,\ P(\chi_n = s_r)) \tag{2.5}$$

als Zustandsverteilung nach n Zeitschritten.

Eine speziell ausgezeichnete Zustandsverteilung ist die *Anfangszustandsverteilung*.

Definition 2.8 (Anfangszustandsverteilung einer DTMC)
Angenommen $S = \{s_1,\ s_2,\ \ldots,\ s_r\}$ ist der diskrete Zustandsraum einer DTMC $\{\chi_t\}$. Die Zustandsverteilung nach Null Zeitschritten

$$\pi^0 := (P(\chi_0 = s_1),\ P(\chi_0 = s_2),\ \ldots,\ P(\chi_0 = s_r)) \tag{2.6}$$

wird Anfangszustandsverteilung genannt.

Mittels der Übergangswahrscheinlichkeitsmatrix $\mathbf{P}$ können beliebige Zustandsverteilungen, nach folgendem Berechnungsmuster, ermittelt werden.

$$\begin{aligned}
\pi^0 &\quad \text{wird vorgegeben} \\
\pi^1 &= \pi^0 \cdot \mathbf{P} \\
\pi^2 &= \pi^1 \cdot \mathbf{P} = \pi^0 \cdot \mathbf{P} \cdot \mathbf{P} = \pi^0 \cdot \mathbf{P}^2 \\
&\vdots \\
\pi^n &= \pi^{(n-1)} \cdot \mathbf{P} = \pi^0 \cdot \mathbf{P}^n.
\end{aligned} \tag{2.7}$$

Wie bereits einleitend erwähnt, ist eines der Kernprobleme bei der Behandlung von DTMCs, die Berechnung der *stationären Zustandsverteilung*. Die Beantwortung der vorangestellten Frage, ob überhaupt eine stationäre Zustandsverteilung existiert, hängt im wesentlichen von der Struktur des Zustandsraumes ab. Für eine genauere Betrachtung der Existenzfrage wird auf [2] verwiesen.

Definition 2.9 (Stationäre Zustandsverteilung einer DTMC)
Eine Zustandsverteilung π wird als stationäre Zustandsverteilung bezeichnet, wenn für die Übergangswahrscheinlichkeitsmatrix $\mathbf{P}$ gilt

$$\pi = \pi \cdot \mathbf{P}. \tag{2.8}$$

Die Bedeutung dieser Definition lässt sich anhand der folgenden Implikation erkennen. Dafür setzen wir en Implikation erkennen. Dafür setzen wir $\pi := \pi^{(n)}$ als stationäre Zustandsverteilung für ein beliebiges n an.

$$\pi^{(n+1)} = \pi^{(n)} \cdot \mathbf{P} = \pi \cdot \mathbf{P} = \pi \qquad \Rightarrow \qquad \pi^{(n+k)} = \pi \qquad \forall k \geq 0. \tag{2.9}$$

Sobald also nach n Zeitschritten eine stationäre Zustandsverteilung erreicht ist, verändert sich die Zustandsverteilung, für beliebig viele weitere Zeitschritte, nicht mehr. Wir können daraus ableiten, dass wenn man das System nach einer gewissen Einschwungsphase zu einem willkürlichen Zeitpunkt betrachtet, ist die Wahrscheinlichkeit, dass es sich in einem Zustand $s_i \in S$ befindet immer gleich.

Nach der Definition der *Identitätsmatrix*, soll auf die Berechnung der stationären Zustandsverteilung eingegangen werden.

Definition 2.10 (Identitätsmatrix I)
*Eine reelle $(r \times r)$-Matrix **I** wird als Identitätsmatrix bezeichnet, wenn ihre Diagonalelemente alle eins und die restlichen Matrixelemente Null sind.*

Es gilt

$$\begin{aligned} \pi = \pi \cdot \mathbf{P} &\Leftrightarrow \pi - \pi \cdot \mathbf{P} = 0 \\ &\Leftrightarrow \pi\left(\mathbf{I} - \mathbf{P}\right) = 0. \end{aligned}$$

Wir folgern, dass die gesuchte Verteilung, mittels des linearen Gleichungssystems

$$\pi\left(\mathbf{I} - \mathbf{P}\right) = 0, \qquad \text{wobei} \qquad \sum_{i=1}^{r} \pi\left(s_i\right) = 1, \tag{2.10}$$

ermittelt werden kann.

Beispiel 2 (Netzwerk 1) *Wir setzen das eingeführte Beispielnetzwerk 1 fort und berechnen beispielhaft verschiedene Zustandsverteilungen, sowie eine angenäherte stationäre Verteilung.*

$$\begin{aligned} \pi^0 &= \left(P\left(\chi_0 = s_1\right),\ P\left(\chi_0 = s_2\right),\ \ldots,\ P\left(\chi_0 = s_6\right)\right) \\ &= (1.0,\ 0,\ 0,\ 0,\ 0,\ 0) \end{aligned}$$

Da der Zustand s_1 der ausgezeichnete Startzustand des Netzwerkes ist, haben wir 100% Wahrscheinlichkeit, dass sich das System nach Null Zeitschritten im Zustand s_1 befindet. Im allgemeinen Fall kann eine willkürliche Zustandsverteilung für π^0 vereinbart werden.

$$\begin{aligned} \pi^1 &= \pi^0 \cdot \mathbf{P} = (1.0,\ 0,\ 0,\ 0,\ 0,\ 0) \cdot \begin{pmatrix} 0.8 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0.1 & 0 & 0.3 & 0 \\ 0 & 0 & 0.4 & 0.6 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0.4 \\ 0 & 0.3 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0.9 \end{pmatrix} \\ &= (0.8,\ 0.2,\ 0,\ 0,\ 0,\ 0) \end{aligned}$$

Das Ergebnis der Berechnung ist bereits bekannt. Es korrespondiert mit der Einschritt-Übergangswahrscheinlichkeit für den Zustand s_1 als Ausgangszustand. Des weiteren gilt

$$\begin{aligned} \pi^2 &= \pi^1 \cdot \mathbf{P} \\ &= (0.8,\ 0.2,\ 0,\ 0,\ 0,\ 0) \cdot \mathbf{P} \\ &= (0.64,\ 0.28,\ 0.02,\ 0.06,\ 0,\ 0)\,. \end{aligned}$$

Wir erkennen an dieser Verteilung, dass sich unser System nach 2 Zeitschritten mit der Wahrscheinlichkeit von 64% noch in Zustand s_1 befindet. Mit der Wahrscheinlichkeit

von 28% wird es nach s_2, mit der Wahrscheinlichkeit von 2% wird es nach s_3 oder mit der Wahrscheinlichkeit von 6% nach s_4 wechseln.

Um die stationäre Verteilung π zu erhalten, müssen wir das folgende lineare Gleichungssystem lösen.

$$
\begin{array}{rrrrrrr}
0.2 \cdot \pi_{s_1} + & -0.2 \cdot \pi_{s_2} & & & & = & 0 \\
0.4 \cdot \pi_{s_2} + & -0.1 \cdot \pi_{s_3} & + & -0.3 \cdot \pi_{s_5} & = & 0 \\
0.6 \cdot \pi_{s_3} + & -0.6 \cdot \pi_{s_4} & & & = & 0 \\
-0.6 \cdot \pi_{s_3} + & 1 \cdot \pi_{s_4} & + & -0.4 \cdot \pi_{s_6} & = & 0 \\
-0.3 \cdot \pi_{s_2} & + & -0.7 \cdot \pi_{s_4} + & 1 \cdot \pi_{s_5} & = & 0 \\
& & & -0.1 \cdot \pi_{s_5} + & 0.1 \cdot \pi_{s_6} = & 0 \\
\pi_{s_1} + & \pi_{s_2} + & \pi_{s_3} + & \pi_{s_4} + & \pi_{s_5} + & \pi_{s_6} = & 1
\end{array}
$$

Das lineare Gleichungssystem (2.10) liefert keine exakte Lösung. Dies bedeutet, dass nach beliebig vielen Zeitschritten immer noch winzig kleine Veränderungen in den Zustandsverteilungen auftreten können. Wir werden versuchen eine angenäherte stationäre Verteilung, mittels einer relativ hohen Anzahl an Zeitschritten, zu berechnen.

$$
\begin{array}{rrl}
\pi^0 & & = (1,\ 0,\ 0,\ 0,\ 0,\ 0) \\
\pi^1 & = \pi^{(0} \cdot \mathbf{P} & = (0.8,\ 0.2,\ 0,\ 0,\ 0,\ 0) \\
\pi^{10} & = \pi^0 \cdot \mathbf{P^{10}} & \approx (0.107,\ 0.166,\ 0.198,\ 0.174,\ 0.080,\ 0.275) \\
\pi^{100} & = \pi^0 \cdot \mathbf{P^{100}} & \approx (0.203 \cdot 10^{-9},\ 0.056,\ 0.153,\ 0.144,\ 0.074,\ 0.574) \\
\pi^{(10^3)} & = \pi^0 \cdot \mathbf{P^{(10^3)}} & \approx (0.123 \cdot 10^{-96},\ 0.056,\ 0.153,\ 0.144,\ 0.074,\ 0.574) \\
\pi^{(10^4)} & = \pi^0 \cdot \mathbf{P^{(10^4)}} & \approx (0,\ 0.056,\ 0.153,\ 0.144,\ 0.074,\ 0.574) \\
\pi^{(10^5)} & = \pi^0 \cdot \mathbf{P^{(10^5)}} & \approx (0,\ 0.056,\ 0.153,\ 0.144,\ 0.074,\ 0.574) \\
& \Rightarrow \pi & \approx (0,\ 0.056,\ 0.153,\ 0.144,\ 0.074,\ 0.574)
\end{array}
$$

Evaluiert man die angenäherte Lösung $\pi \approx (0,\ 0.056,\ 0.153,\ 0.144,\ 0.074,\ 0.574)$, kann gefolgert werden, dass sich das System im Zustand s_1 mit einer Wahrscheinlichkeit von 0%, im Zustand s_2 mit der Wahrscheinlichkeit von 0.056%, usw. befindet, sobald eine angenäherte stationäre Zustandsverteilung erreicht ist.

Vergleichen wir die berechneten Werte mit dem Aufbau des Zustandsraumes (Abb. (2.3)), lassen sich direkte Korrelationen erkennen:

- Der Zustand s_1 besitzt keine eingehenden Kanten. Daraus kann geschlossen werden, sobald das System den Zustand s_1 erstmals verlässt, ist es ihm nicht mehr möglich s_1 erneut zu erreichen. Damit konvergiert die Aufenthaltswahrscheinlichkeit für Zustand s_1 gegen Null, sobald eine entsprechend große Anzahl an Zeitschritten absolviert wurde.

- Die Zustände s_2, s_3, s_4, s_5 und s_6 befinden sich in derselben terminalen stark zusammenhängenden Komponente des Zustandsgraphen. Dies bedeutet, zwischen jeweils

zwei beliebigen Knoten dieser Menge existiert ein gerichteter Pfad [26]. Befindet sich das System erst einmal in dieser terminalen Komponente, sind die Wahrscheinlichkeiten, dass sich das System in einem dieser Zustände aufhält stets größer Null, da alle Zustände der Komponente immer wieder besucht werden können.

- Befindet sich das System im Zustand s_6, ist die Wahrscheinlichkeit im nächsten Zeitschritte in s_6 zu verharren 90%. Es verwundert also nicht, das die Aufenthaltswahrscheinlichkeit für Zustand s_6 am größten ist.

2.3 Kontinuierliche Markow-Kette

Im vorherigen Abschnitt wurden Markow-Ketten mit diskretem Zeitverlauf untersucht. Die für uns interessanten biochemischen Prozesse laufen jedoch in einem kontinuierlichen Zeitkontext ab. Deshalb wird in diesem Abschnitt eine theoretische Aufarbeitung der *kontinuierlichen Markow-Ketten* präsentiert.

Definition 2.11 (Kontinuierliche Markow-Kette)
Angenommen $(\Omega, \mathcal{F}, \mathrm{P})$ *ist ein Wahrscheinlichkeitsraum. Der Zustandsraum* S *ist abzählbar und die Menge* $T \subseteq \mathbb{R}_+$ *spezifiziert einen Zeitindex. Die Familie von Zufallsvariablen*

$$\{\chi_t : \Omega \to S |\ t \in T\}$$

ist eine kontinuierliche Markow-Kette, falls die folgende kontinuierliche Variante der Markow-Eigenschaft gilt.

$$P\left(\chi_{t_{(n+1)}} = s \mid \chi_{t_n} = s_n, \ldots, \chi_{t_0} = s_0\right) = P\left(\chi_{t_{(n+1)}} = s \mid \chi_{t_n} = s_n\right) \tag{2.10}$$

für alle $n \in \mathbb{N}$, $t_{(n+1)} > t_n > \ldots > t_0$, *alle Folgen* $\left\{t_0, t_1, \ldots, t_{(n+1)}\right\}$ *und alle* $s, s_n, s_{(n-1)}, \ldots, s_0 \in S$ *mit* $P\left(\chi_{t_n} = s_n, \ldots, \chi_{t_0} = s_0\right) > 0$.

Auch für kontinuierliche Markow-Ketten, soll eine abkürzende Schreibweise eingeführt werden. Wir wollen das Akronym CTMC benutzen, welches für die englische Bezeichnung *continuous-time Markov chain* steht.

Die kontinuierliche Markow-Eigenschaft drückt abermals die Idee aus, dass das zukünftige Verhalten nur vom aktuellen Zustand abhängt und nicht zusätzlich von der Vergangenheit.

In den nächsten Unterabschnitten sollen für CTMCs dieselben Eigenschaften und Charakteristika abgeleitet werden, die bereits für DTMCs vorgestellt wurden. Die Tabelle (2.1) stellt die verschiedenen Eigenschaften gegenüber.

2.3.1 Übergangswahrscheinlichkeiten einer CTMC

Wenn sich der zu untersuchende Prozess im Zeitpunkt w in einem Zustand s_i befindet, kann die Wahrscheinlichkeit für einen Zustandsübergang, innerhalb des nächsten Zeitintervalles ti, mithilfe der *Zeitintervall-Übergangswahrscheinlichkeit* beschrieben werden.

Definition 2.12 (Zeitintervall-Übergangswahrscheinlichkeit einer CTMC)
Angenommen $S = \{s_1, s_2, \ldots, s_r\}$ ist der Zustandsraum einer CTMC $\{\chi_t\}$. Die Wahrscheinlichkeit

$$P(s_i, s_j, w, ti) := P\left(\chi_{t_{(w+ti)}} = s_j \mid \chi_{t_w} = s_i\right), \qquad s_i, s_j \in S \tag{2.11}$$

ist als die Zeitintervall-Übergangswahrscheinlichkeit für das Zeitintervall ti definiert. Ist diese Wahrscheinlichkeit unabhängig vom Zeitpunkt w, wird der Prozess als zeithomogen angesehen und man kann in diesem Fall die Zeitintervall-Übergangswahrscheinlichkeit in der vereinfachten Form $P(s_i, s_j, ti)$ angeben.

Es wird im weiteren von zeithomogenen CTMCs ausgegangen. Abermals werden die einzelnen Übergangswahrscheinlichkeiten in einer entsprechenden Matrix zusammen gefasst.

Definition 2.13 (Zeitintervall-Übergangsmatrix)
Angenommen $S = \{s_1, s_2, \ldots, s_r\}$ ist der Zustandsraum einer CTMC $\{\chi_t\}$. Die folgende Matrix aus einzelnen Zeitintervall-Übergangswahrscheinlichkeiten für ein Zeitintervall ti wird als Zeitintervall-Übergangsmatrix für das Zeitintervall ti bezeichnet.

$$\mathbf{P_{ti}} := \begin{pmatrix} P(s_1, s_1, ti) & \ldots & P(s_1, s_r, ti) \\ \vdots & \ddots & \vdots \\ P(s_r, s_1, ti) & \ldots & P(s_r, s_r, ti) \end{pmatrix} \tag{2.12}$$

Die Matrix $\mathbf{P_{ti}}$ ist eine stochastische Matrix. Die Matrix $\mathbf{P_0}$ für das Zeitintervall Null wird auf die Identitätsmatrix $\mathbf{I}$ gesetzt.

Wir besitzen nun ein kontinuierliches Äquivalent zur diskreten Übergangsmatrix $\mathbf{P}$. Jedoch ist es nahezu unmöglich die Menge aller Zeitintervall-Übergangsmatrizen handzuhaben. Es wäre demnach dienlich eine Art Generatormatrix für die automatische Berechnung der Zeitintervallmatrizen zu besitzen.

In der nächsten Definition werden wir die *Übergangsratenmatrix* $\mathbf{Q}$ spezifizieren, die genau diese Generatoreigenschaft mitbringt.

Definition 2.14 (Übergangsratenmatrix einer CTMC)
Angenommen $\mathbf{P_{ti}}$ ist die Zeitintervall-Übergangsmatrix für das Zeitintervall ti einer CTMC $\{\chi_t\}$. Die Matrix

$$\mathbf{Q} := \frac{d}{dti}\mathbf{P_{ti}}|_{ti=0} \tag{2.13}$$

$$= \lim_{\Delta ti \to 0} \frac{\mathbf{P_{\Delta ti}} - \mathbf{P_0}}{\Delta ti} \tag{2.14}$$

$$= \lim_{\Delta ti \to 0} \frac{\mathbf{P_{\Delta ti}} - \mathbf{I}}{\Delta ti} \tag{2.15}$$

wird als Übergangsratenmatrix einer CTMC definiert.

Es soll darauf hingewiesen werden, dass die Übergangsratenmatrix $\mathbf{Q}$ keine stochastische Matrix ist, wie die vorher eingeführten Matrizen $\mathbf{P}$ und $\mathbf{P_{ti}}$. A priori geben die Elemente der Übergangsratenmatrix $\mathbf{Q}$ zunächst lediglich Quantitäten für die Zustandswechsel an. Wir werden eine erste Interpretation nach der Definition der *infinitesimalen Übergangsmatrix* kennen lernen. Diese theoretische Hilfsgröße beinhaltet die Übergangswahrscheinlichkeiten für ein unendlich kleines Zeitintervall dti.

Definition 2.15 (Infinitesimale Übergangsmatrix einer CTMC)
Angenommen $\mathbf{Q}$ *ist die Übergangsratenmatrix einer CTMC* $\{\chi_t\}$*. Die Matrix*

$$\mathbf{P_{dti}} := \mathbf{I} + \mathbf{Q} \cdot dti \tag{2.16}$$

ist die infinitesimale Übergangsmatrix einer CTMC. Die Matrix $\mathbf{P_{dti}}$ *ist eine stochastische Matrix.*

Aufgrund der Eigenschaften von stochastischen Matrizen (Def. (2.6)) und dem Zusammenhang zwischen $\mathbf{Q}$ und $\mathbf{P_{dti}}$ (Def. (2.15)) ergeben sich folgende Nebenbedingungen für das Aufstellen der Übergangsratenmatrix $\mathbf{Q}$:

- die Nicht-Diagonalelemente von $\mathbf{P_{dti}}$ und $\mathbf{Q}$ müssen gleich sein,
- die Diagonalelemente von $\mathbf{Q}$ dürfen nicht positiv sein und
- die Zeilen von $\mathbf{Q}$ summieren sich zu Null.

Daraus folgt direkt, dass sich jedes Diagonalelement als negierte Summe der Nicht-Diagonalelemente derselben Zeile ergibt.

$$q[i,i] = (-1 \cdot \sum_{(j=1, j\neq i)}^{r} q[i,j]) \tag{2.17}$$

Wenn wir die Matrixelemente der Übergangsratenmatrix $\mathbf{Q}$, bezüglich der Definition (2.15) einzeln ausschreiben, erhalten wir die beiden Gleichungen

$$q[i,j] = \lim_{\Delta ti \to 0} \frac{p_{\Delta ti}[i,j]}{\Delta ti}, \qquad für\ i \neq j, \tag{2.18}$$

$$q[i,i] = \lim_{\Delta ti \to 0} \frac{p_{\Delta ti}[i,i] - 1}{\Delta ti}. \tag{2.19}$$

Eine erste intuitive Interpretation für die Matrixelemente von $\mathbf{Q}$, welche auf der Gleichung (2.16) basiert, kann wie folgt formuliert werden. Befindet sich das betrachtete System im Zustand s_i, ist die angenäherte Wahrscheinlichkeit, dass ein Zustandsübergang, innerhalb des nächsten Zeitintervalles Δti zu einem Zustand s_j, statt findet gleich $(q[i,j] \cdot \Delta ti)$ ist. Dementsprechend quantifiziert $(-q[i,i] \cdot \Delta ti)$ die angenäherte Wahrscheinlichkeit, dass das System den momentanen Zustand s_i, innerhalb des Zeitintervalles Δti, verlässt. Für kleine Zeitintervalle Δti können wir diese Interpretation aus der Approximation

$$\mathbf{P_{dti}} = \mathbf{I} + \mathbf{Q} \cdot dti \tag{2.20}$$

$$\Rightarrow \mathbf{P_{\Delta ti}} \approx \mathbf{I} + \mathbf{Q} \cdot \Delta ti \tag{2.21}$$

ableiten. Im nächsten Unterabschnitt wird eine weitere Eigenschaft eines Matrixelementes von $\mathbf{Q}$ vorgestellt.

Neben der Approximation (2.21) liefert uns die Hilfsgröße der infinitesimalen Übergangsmatrix $\mathbf{P_{dti}}$ eine Methode zur direkten Berechnung von $\mathbf{P_{ti}}$. Dafür betrachten wir die Ableitung von $\mathbf{P_{ti}}$ für ein willkürliches Zeitintervall ti.

$$\begin{aligned}\frac{d}{dti}\mathbf{P_{ti}} &= \frac{\mathbf{P_{(ti+dti)}} - \mathbf{P_{ti}}}{dti} \\ &= \frac{(\mathbf{P_{ti}} \cdot \mathbf{P_{dti}}) - \mathbf{P_{ti}}}{dti} \\ &= \frac{\mathbf{P_{dti}} - \mathbf{I}}{dti} \cdot \mathbf{P_{ti}} \\ &= \mathbf{Q} \cdot \mathbf{P_{ti}}\end{aligned} \tag{2.22}$$

Folglich ist $\mathbf{P_{ti}}$ eine Lösung der Differentialgleichung

$$\frac{d}{dti}\mathbf{P_{ti}} = \mathbf{Q} \cdot \mathbf{P_{ti}} \tag{2.23}$$

mit dem Anfangswert $\mathbf{P_0} = \mathbf{I}$. Diese Gleichung hat die Lösung

$$\mathbf{P_{ti}} = \exp\{\mathbf{Q} \cdot \mathbf{P_{ti}}\}, \tag{2.24}$$

wobei $\exp\{\cdot\}$ der Matrizenexponentialfunktion entspricht [32].

2.3.2 Aufenthaltsdauer und Wartezeit

Bis jetzt wurden vornehmlich Wahrscheinlichkeiten für Zustandsübergänge zu diskreten Zeitpunkten oder innerhalb kontinuierlicher Zeitintervalle diskutiert. Der Betrachtungsgegenstand soll nun geändert werden. Wir wollen uns weg von Wahrscheinlichkeiten für Zustandswechsel, hin zu Zeitabständen zwischen den Zustandsübergängen, bewegen. Wir werden bei diesem Vorgehen die Begriffe der *Aufenthaltsdauer in einem Zustand* und der *Wartezeit, bis ein bestimmter Zustandswechsel auftritt,* entwickeln.

Zwei wichtige Implikationen der Markow-Eigenschaft, die jeweils in einer diskreten und einer kontinuierlichen Ausprägung (Def. (2.2) und (2.11)) eingeführt wurden, können wie folgt fest gehalten werden.

- Die erwartete Aufenthaltsdauer in einem Zustand s_i, bis ein beliebiger Zustandswechsel eintritt, und
- die Wartezeit, welche verstreicht bis ein Zustandswechsel zwischen zwei bestimmten Zuständen s_i und s_j, statt findet,

sind exponentiell verteilt [2]. Die Exponentialverteilung ist die einzige Wahrscheinlichkeitsverteilung, welche die Bedingung der Gedächtnislosigkeit für die kontinuierliche Markow-Eigenschaft erfüllt.

Die folgenden beiden Definitionen benutzen die Elemente $q[i,i]$ und $q[i,j]$ der Übergangsratenmatrix $\mathbf{Q}$, die in der Gleichung (2.19) explizit formuliert wurden.

Definition 2.16 (Aufenthaltsdauer in einem Zustand einer CTMC)
Angenommen $S = \{s_1, s_2, \ldots, s_r\}$ ist der Zustandsraum und $\mathbf{Q}$ ist die Übergangsratenmatrix einer CTMC $\{\chi_t\}$. Die Aufenthaltsdauer in einem Zustand $s_i \in S$ bis ein beliebiger Zustandswechsel auftritt, ist eine exponentiell verteilte Zufallsvariable SJ_{s_i}. Die Dichtefunktion der Zufallsvariable SJ_{s_i} ist gegeben durch

$$f_{SJ_{s_i}}(ti) = -q[i,i] \cdot \mathrm{e}^{(q[i,i] \cdot ti)}, \qquad ti \geq 0. \tag{2.25}$$

Folglich kann die mittlere Aufenthaltsdauer in einem Zustand $s_i \in S$, mit dem Erwartungswert $E[SJ_{s_i}] = -\frac{1}{q[i,i]}$ der Zufallsvariable SJ_{s_i}, bestimmt werden.

Die Wartezeit für einen Zustandswechsel, zwischen einem bestimmten Paar von Zuständen, wird ebenfalls auf der Grundlage der Matrixelemente von $\mathbf{Q}$ definiert.

Definition 2.17 (Wartezeit bis ein bestimmter Zustandswechsel auftritt)
Angenommen $S = \{s_1, s_2, \ldots, s_r\}$ ist der Zustandsraum und $\mathbf{Q}$ ist die Übergangsratenmatrix einer CTMC $\{\chi_t\}$. Die Wartezeit bis ein Zustandswechsel zwischen den Zuständen s_i und s_j auftritt, ist eine exponentiell verteilte Zufallsvariable $ET_{(s_i,s_j)}$. Die Dichtefunktion der Zufallsvariable $ET_{(s_i,s_j)}$ ist gegeben durch

$$f_{ET_{(s_i,s_j)}}(ti) = q[i,j] \cdot \mathrm{e}^{(-q[i,j] \cdot ti)}, \qquad ti \geq 0. \tag{2.26}$$

Abermals kann ein Erwartungswert $E[ET_{(s_i,s_j)}] = \frac{1}{q[i,j]}$ berechnet werden. In diesem Fall erhalten wir die mittlere Zeit bis das System, ausgehend von Zustand s_i, in den Zustand s_j wechselt. Demnach drückt $q[i,j]$ die Übergangsrate aus, wie oft dieser Zustandsübergang durchschnittlich in einer Zeiteinheit vollzogen wird.

Es soll betont werden, dass es sich bei den betrachteten Zeitintervallen um echte Wartezeiten handelt. Der eigentliche Zustandswechsel wird als zeitlos betrachtet.

Zusammenfassend kann fest gehalten werden, dass das Aufstellen der Übergangsratenmatrix $\mathbf{Q}$ genügt, um das gesamte dynamische Verhalten einer CTMC zu bestimmen. Die Elemente der Matrix $\mathbf{Q}$ werden dabei als Übergangsraten für Zustandsübergänge charakterisiert.

Die Diagonalelemente $q[i,i]$ von $\mathbf{Q}$ können dabei aus den bereits modellierten Übergangsraten derselben Zeile abgeleitet werden, siehe Gleichung (2.17).

Um den gesamten stochastischen Prozess eindeutig bestimmen zu können, benötigten wir zusätzlich eine Anfangskonfiguration des Systems, in Form einer Anfangszustandsverteilung.

Beispiel 3 (Netzwerk 2) *Wir werden das Beispielnetzwerk aus dem vorherigen Kapitel als kontinuierliche Variante untersuchen. Im Gegensatz zum vorangegangen Ansatz modellieren wir hier nicht mit Übergangswahrscheinlichkeiten, sondern mit Übergangsraten. Der diskrete Zustandsraum verändert sich nicht und ist abermals $S = \{s_1, \ldots, s_6\}$. Angenommen wir spezifizieren die Übergangsratenmatrix $\boldsymbol{Q}$ als*

$$\mathbf{Q} := \begin{pmatrix} -1.20 & 1.20 & 0 & 0 & 0 & 0 \\ 0 & -1.85 & 1.15 & 0 & 0.70 & 0 \\ 0 & 0 & -2.60 & 2.60 & 0 & 0 \\ 0 & 0 & 2.60 & -5.00 & 0 & 2.40 \\ 0 & 0.70 & 0 & 1.70 & -2.40 & 0 \\ 0 & 0 & 0 & 0 & 1.11 & -1.11 \end{pmatrix}$$

.

So ist zum Beispiel die Übergangsrate von Zustand s_1 in Zustand s_2 1.20. Dies bedeutet, dass durchschnittlich 1.20 Zustandswechsel in einer Zeiteinheit auftreten. Oder anders ausgedrückt, es vergehen durchschnittlich $\frac{5}{6}$ Zeiteinheiten bis das System, ausgehend von Zustand s_1, den Zustand s_2 einnimmt. Wie im diskreten Fall, ist das Bestimmen dieser Quantitäten einer der zentralen Modellierungsschritte.

Mithilfe der soeben aufgestellten Matrix $\mathbf{Q}$ berechnen wir die zwei Zeitintervall-Übergangsmatrizen $\mathbf{P_{0.05}}$ und $\mathbf{P_{0.15}}$. Wir nutzen dazu die Approximation (2.21), wobei Δti jeweils 0.05 *und* 0.15 *sind.*

$$\mathbf{P_{0.05}} \approx \mathbf{I} + \mathbf{Q} \cdot 0.05 \approx \begin{pmatrix} 0.940 & 0.060 & 0 & 0 & 0 & 0 \\ 0 & 0.907 & 0.058 & 0 & 0.035 & 0 \\ 0 & 0 & 0.870 & 0.130 & 0 & 0 \\ 0 & 0 & 0.130 & 0.750 & 0 & 0.120 \\ 0 & 0.035 & 0 & 0.085 & 0.88 & 0 \\ 0 & 0 & 0 & 0 & 0.055 & 0.945 \end{pmatrix}$$

$$\mathbf{P_{0.15}} \approx \mathbf{I} + \mathbf{Q} \cdot 0.15 \approx \begin{pmatrix} 0.820 & 0.180 & 0 & 0 & 0 & 0 \\ 0 & 0.722 & 0.173 & 0 & 0.105 & 0 \\ 0 & 0 & 0.610 & 0.390 & 0 & 0 \\ 0 & 0 & 0.390 & 0.250 & 0 & 0.360 \\ 0 & 0.105 & 0 & 0.255 & 0.640 & 0 \\ 0 & 0 & 0 & 0 & 0.167 & 0.833 \end{pmatrix}$$

Abschließend greifen wir den bereits aus dem diskreten Beispiel bekannten Simulationslauf run_1 auf und berechnen dessen Wahrscheinlichkeit unter der vereinfachten Annahme, dass ein Zustandswechsel nur aller 0.05 oder 0.15 Zeitschritte erfolgt.

$$\begin{aligned} & s_1 \overset{0.05}{\rightarrow} s_1 \overset{0.05}{\rightarrow} s_2 \overset{0.15}{\rightarrow} s_5 \overset{0.15}{\rightarrow} s_4 \overset{0.05}{\rightarrow} s_6 \\ run_1' := & (\chi_{(0,0.05]} = s_1, \chi_{(0.05,1]} = s_1, \chi_{(0.1,0.25]} = s_2, \chi_{(0.25,0.4]} = s_5, \\ & \chi_{(0.4,0.45]} = s_4, \chi_{(0.45,\infty)} = s_6) \\ P(run_1') \approx & P(s_1,\ s_1,\ 0.05) \cdot P(s_1,\ s_2,\ 0.05) \cdot P(s_2,\ s_5,\ 0.15) \cdot \\ & P(s_5,\ s_4,\ 0.15) \cdot P(s_4,\ s_6,\ 0.05) \\ \approx & 0.940 \cdot 0.060 \cdot 0.105 \cdot 0.255 \cdot 0.120 \\ \approx & 0.00018 \end{aligned}$$

DTMC		**CTMC**	
Einschritt-Übergangs-wahrscheinlichkeit	$P(s_x, s_y)$	Zeitintervall-Übergangs-wahrscheinlichkeit	$P(s_x, s_y, ti)$
Übergangsmatrix	$\mathbf{P}$	Zeitintervall-Übergangsmatrix, Infinitesimale Übergangsmatrix, Übergangsratenmatrix	$\mathbf{P_{ti}}$, $\mathbf{P_{dti}}$, $\mathbf{Q}$
Stationäre Verteilung	π	Stationäre Verteilung	π

Tabelle 2.1: Beziehungen zwischen DTMCs und CTMCs.

2.3.3 Stationäre Zustandsverteilung

Die infinitesimale Übergangsmatrix $\mathbf{P_{dti}}$ bietet uns ebenfalls die Möglichkeit eine stationäre Zustandsverteilung für die kontinuierliche Variante der Markow-Ketten zu definieren.

Definition 2.18 (Stationäre Zustandsverteilung einer CTMC)
Angenommen $\mathbf{P_{dti}}$ *ist die infinitesimale Übergangsmatrix einer CTMC* $\{\chi_t\}$. *Eine Zustandsverteilung* π *wird als stationäre Zustandsverteilung einer CTMC bezeichnet, wenn gilt*

$$\pi = \pi \cdot \mathbf{P_{dti}}. \tag{2.27}$$

Wie bereits für den diskreten Fall sind wir damit in der Lage ein lineares Gleichungssystem abzuleiten, aus welchem die stationäre Zustandsverteilung π berechnet werden kann, falls diese existiert.

$$\begin{aligned} & \pi \cdot \mathbf{P_{dti}} && = \pi \\ \Rightarrow \quad & \pi \cdot (\mathbf{I} + \mathbf{Q} \cdot dti) && = \pi \\ \Rightarrow \quad & \pi + \pi \cdot \mathbf{Q} && = \pi \\ \Rightarrow \quad & \pi \cdot \mathbf{Q} && = 0 \end{aligned}$$

Hierbei wird $\mathbf{Q} \cdot dti$ durch $\mathbf{Q}$ ersetzt, da das winzig kleine Zeitanteil dti vernachlässigt werden kann. Zusammenfassend erhalten wir schließlich

$$\pi \cdot \mathbf{Q} = 0, \qquad \text{wobei} \qquad \sum_{i=1}^{r} \pi\left(s_i\right) = 1. \tag{2.28}$$

Beispiel 4 (Netzwerk 2) *Auch für die kontinuierliche Variante unseres Beispielnetzwerkes besitzt das Gleichungssystem (2.28) keine eindeutige Lösung. Daraus kann man abermals schließen, dass es nach beliebig langen Zeitintervallen, noch zu Veränderungen in den Zustandsverteilungen kommt. Wir werden jedoch dieselbe Approximationstechnik anwenden, die bereits im diskreten Fall eine verwertbare angenäherte Lösung ermittelt hat. Wir substituieren hierbei die infinitesimale Übergangsmatrix* $\mathbf{P_{dti}}$ *durch die bereits*

berechnete Zeitintervall-Übergangsmatrix $\mathbf{P_{0.05}}$ *und wenden anschließend die Gleichung* $\pi^n = \pi^0 \cdot \mathbf{P^n_{dti}}$ *an, welche die kontinuierliche Version der Gleichung (2.7) ist.*

$$
\begin{array}{rrl}
\pi^0 & & = (1,\ 0,\ 0,\ 0,\ 0,\ 0) \\
\pi^1 & \approx \pi^0 \cdot \mathbf{P_{0.05}} & \approx (0.940,\ 0.060,\ 0,\ 0,\ 0,\ 0) \\
\pi^{100} & \approx \pi^0 \cdot \mathbf{P^{100}_{0.05}} & \approx (0.205 \cdot 10^{-2},\ 0.074,\ 0.203,\ 0.170,\ 0.188,\ 0.363) \\
\pi^{(10^5)} & \approx \pi^0 \cdot \mathbf{P^{(10^5)}_{0.05}} & \approx (0,\ 0.072,\ 0.201,\ 0.170,\ 0.194,\ 0.366) \\
\pi^{(10^6)} & \approx \pi^0 \cdot \mathbf{P^{(10^6)}_{0.05}} & \approx (0,\ 0.072,\ 0.201,\ 0.170,\ 0.194,\ 0.366) \\
& \Rightarrow \pi & \approx (0,\ 0.072,\ 0.201,\ 0.170,\ 0.194,\ 0.366)
\end{array}
$$

Die Wahrscheinlichkeitsmasse fließt von Zustand s_1 *zur Zustandsmenge* $\{s_2, \ldots, s_6\}$. *Der Zustand* s_6 *besitzt die größte Aufenthaltswahrscheinlichkeit. Die letztere Beobachtung wird durch den Vergleich der Eingangs- und Ausgangsraten aller Zustände der stark zusammenhängenden Komponente* $\{s_2, \ldots, s_6\}$ *bestätigt. Der Zustand* s_6 *besitzt, mit dem Wert 1.29, die größte Differenz und damit den höchsten Netto-Zufluss.*

Kapitel 3

Stochastische Petrinetze

3.1 Kontinuierliche Massenwirkungskinetik

Bevor im anschließenden Abschnitt die *stochastische Massenwirkungskinetik* vorgestellt wird, wollen wir eine kurze Einführung in ihr deterministisches Gegenstück geben. Die *deterministische Massenwirkungskinetik* stellt einen weit verbreiteten klassischen Ansatz zur Modellierung und Simulation von biochemischen Netzwerken dar.

Netzwerke, die mittels der deterministischen Massenwirkungskinetik modelliert werden, benutzen eine kontinuierliche Konzentration, gemessen in *Mol*, für die Angabe der einzelnen aktuellen Stoffmengen. Die Teilchenanzahl pro Mol, die mit der so genannten Avogadro-Konstante ausgedrückt wird, beträgt ungefähr $6.022 \cdot 10^{23}$. Eine Konzentration des Stoffes A von $[A] = 0.2341$ würde demnach circa $1.410 \cdot 10^{23}$ Teilchen entsprechen [7].

Die Geschwindigkeit, mit der Stoffe in den *deterministischen Massenwirkungsreaktionen* umgesetzt werden, ist direkt proportional zu den momentanen Stoffkonzentrationen und den stöchiometrischen Koeffizienten der Reaktionen. Diese Gesetzmäßigkeit ist der zentrale Grundsatz der deterministischen Massenwirkungskinetik.

Jede deterministische Massenwirkungsreaktion kann demnach durch einen festen Proportionalitätsfaktor charakterisiert werden. Diese feste Kennzahl wird als *deterministische Reaktionskonstante* k_j definiert. Sie ist nicht mit der gegenwärtigen *Transformationsrate* v_j zu verwechseln, welche sich in einem laufenden Reaktionsvorgang ständig verändern kann.

Wir werden im folgenden diese Konzepte und deren Beziehungen definieren und untersuchen.

Definition 3.1 (Elementare deterministische Massenwirkungsreaktion)
Es wird ein biochemisches Netzwerk betrachtet, an dem u Stoffe beteiligt sind. Es besteht insgesamt aus w unidirektionalen elementaren deterministische Massenwirkungsreaktionen. Eine einzelne elementare Reaktion kann dann beschrieben werden als:

$$R_j : \qquad \sum_{i=1}^{u} l_{(i,j)} \cdot A_i \xrightarrow{k_j} \sum_{i=1}^{u} r_{(i,j)} \cdot A_i, \qquad j = 1, \ldots, w, \tag{3.1}$$

wobei A_i der i-te beteiligte Stoff des Netzwerkes und R_j die j-te elementare Reaktion ist. Die Faktoren $l_{(i,j)}$ und $r_{(i,j)}$ sind die stöchiometrischen Koeffizienten der Reaktion R_j. Die Konstante k_j ist die deterministische Reaktionskonstante für R_j.

Beispiel 5 (Netzwerk 3) *Wir betrachten ein biochemisches Netzwerk, welches die Stoffe A, B, C und D umfasst. Diese sind an den determintchen Massenwirkungsreaktionen $R_1, \ldots, R_4$ beteiligt.*

$$R_1 \quad : \qquad 2A + B \xrightarrow{k_1} C \tag{3.2}$$

$$R_2 \quad : \qquad B \xrightarrow{k_2} D \tag{3.3}$$

$$R_3 \quad : \qquad D \xrightarrow{k_3} B \tag{3.4}$$

$$R_4 \quad : \qquad C + 3D \xrightarrow{k_4} 2A \tag{3.5}$$

Wir wollen darauf hinweisen, das die Reaktionen R_2 und R_3 auch durch eine einzelne reversible Reaktion modelliert werden könnten. Da wir jedoch aus formalen Gründen alle Reaktionen als elementar eingeführt haben, werden diese getrennt aufgeführt.

Zur Bestimmung der aktuellen Transformationsrate v_j einer Reaktion R_j müssen wir das *deterministische Massenwirkungsgesetz* anwenden.

Definition 3.2 (Transformationsrate einer det. Massenwirkungsreaktion) *Die momentane Transformationsrate $v_j : [A_i]^u \to \mathbb{R}$ einer deterministischen Massenwirkungsreaktion R_j ist*

$$v_j(([A_1], \ldots, [A_u])) := k_j \cdot \prod_{i=1}^{u} [A_i]^{l_{(i,j)}}. \tag{3.6}$$

Beispiel 6 (Netzwerk 3) *Wir leiten die aktuellen Transformationsraten für unser Beispielnetzwerk ab.*

$$\begin{aligned} v_1(([A], \ldots, [D])) &= k_1 \cdot [A]^2 \cdot [B] \\ v_2(([A], \ldots, [D])) &= k_2 \cdot [B] \\ v_3(([A], \ldots, [D])) &= k_3 \cdot [D] \\ v_4(([A], \ldots, [D])) &= k_4 \cdot [C] \cdot [D]^3 \end{aligned}$$

Da wir mit elementaren Reaktionen arbeiten, muss einer der Faktoren $l_{(i,j)}$ oder $r_{(i,j)}$ immer Null sein. Dies bedeutet, dass ein Stoff in einer elementaren Reaktion entweder Reaktant oder Produkt ist.

Wir sind nun in der Lage ein System von gewöhnlichen Differentialgleichungen (DGL) aufzustellen, dass das kontinuierliche Änderungsverhalten der Stoffkonzentrationen aufzeigt.

Definition 3.3 (DGL-System für det. Massenwirkungsreaktionen) *Angenommen wird ein biochemisches Netzwerk mit deterministischer Massenwirkungskinetik. Es enthält u Stoffe und w elementare Reaktionen. Das System von DGL, welches das Änderungsverhalten der einzelnen Stoffmengen beschreibt, kann wie folgt aufgestellt werden:*

$$\frac{d}{dt}[A_i] = \sum_{j=1}^{w} (l_{(i,j)} - r_{(i,j)}) \cdot v_j(([A_1], \ldots, [A_u])), \qquad i = 1, \ldots, u. \tag{3.7}$$

Beispiel 7 (Netzwerk 3) *Die Differentialgleichungen des Beispielnetzwerkes 3 sind*

$$\begin{aligned}
\frac{d}{dt}[A] &= -2 \cdot v_1 + 2 \cdot v_4, \\
\frac{d}{dt}[B] &= v_1 - v_2 + v_3, \\
\frac{d}{dt}[C] &= v_1 - v_4, \\
\frac{d}{dt}[D] &= v_2 - v_3 - 3 \cdot v_4.
\end{aligned}$$

Eingesetzt ergibt das

$$\begin{aligned}
\frac{d}{dt}[A] &= -2 \cdot k_1 \cdot [A]^2 \cdot [B] + 2 \cdot k_4 \cdot [C] \cdot [D]^3, \\
\frac{d}{dt}[B] &= k_1 \cdot [A]^2 \cdot [B] - k_2 \cdot [B] + k_3 \cdot [D], \\
\frac{d}{dt}[C] &= k_1 \cdot [A]^2 \cdot [B] - k_4 \cdot [C] \cdot [D]^3, \\
\frac{d}{dt}[D] &= k_2 \cdot [B] - k_3 \cdot [D] - 3 \cdot k_4 \cdot [C] \cdot [D]^3.
\end{aligned}$$

3.2 Stochastische Massenwirkungskinetik

Es gibt eine Vielzahl von intra-zellulären Prozessen, die auf Basis einer sehr kleinen Anzahl von Molekülen ablaufen. Gillespie zeigt in [14], dass der deterministische Modellierungsansatz nicht in der Lage ist, die eigentliche Natur dieser biochemischen Netzwerke hinreichend gut darzustellen.

Um die Wechselwirkungen, zwischen deterministischen und stochastischen Modellierungskonzepten, besser aufzeigen zu können, wollen wir drei Modellierungsebenen betrachten:

(i) Verhalten eines einzelnen Moleküls

(ii) Verhalten eines Systems mit wenigen Molekülen pro Stoff

(iii) Verhalten eines Systems mit sehr vielen Molekülen pro Stoff

Wir werden sehen, dass eine abwechselnde Approximation zwischen den Ebenen statt findet. So werden deterministische Konzepte durch stochastische vereinfacht und umgekehrt.

Betrachtet man ein einzelnes Molekül, lässt sich deren Bewegung in einem Modell deterministisch, mittels den Gesetzmäßigkeiten der klassischen Mechanik, vorhersagen.

Aufgrund des hohen Berechnungsaufwandes, der notwendig wäre um das deterministische Verhalten jedes Moleküls zu ermitteln, ist man zu einer stochastischen Modellierung übergegangen.

So wird das erwartete charakteristische Verhalten eines einzelnen Moleküles, mithilfe von stochastischen Prozessen beschrieben. Das tatsächlich beobachtbare Verhalten eines Moleküls variiert dabei, in wohldefinierter Form, um den spezifizierten Erwartungswert. So wird zum Beispiel, zur Modellierung der Brownschen Molekularbewegung, der Wiener-Prozess [27] benutzt.

Die logische Erweiterung dieses Ansatzes ist die Modellierung ganzer Reaktionssysteme auf Basis stochastischer Prozesse. Als weit verbreitetes Beispiel haben wir bereits die kontinuierlichen Markow-Ketten kennen gelernt. Für sehr große Molekülanzahlen ist die Simulation der stochastischen Prozesse nicht mehr effizient durchführbar, siehe Abschnitt (4.2).

Glücklicherweise kann als Ausweg ein Approximationsansatz angewendet werden, den uns die Wahrscheinlichkeitstheorie bereit stellt. Bei entsprechender Interpretation des *Gesetzes der großen Zahlen* [5], nähert sich das auftretende mittlere Verhalten einer sehr großen homogenen Population, dem erwarteten Verhalten eines einzelnen Individuums an. Dies bedeutet in unserem Kontext, dass im Gesamtverhalten einer sehr großen Menge von gleichartigen Molekülen, die stochastischen Abweichungen eines einzelnes Moleküls vernachlässigbar sind.

Die Konsequenz dieser Approximation ist der deterministische Modellierungsansatz, den wir im vorangegangenen Kapitel vorgestellt haben. Dort werden in die Betrachtung keine stochastische Effekte mehr miteinbezogen. Das spezifische Verhalten der Stoffmengen wird nur noch über deterministische Ratenkonstanten beschrieben und es kann zu einem kontinuierlichen Stoffaustausch übergegangen werden.

Wir werden in diesem Abschnitt einen Schritt zurück gehen und eine Möglichkeit vorstellen, mittels derer biochemische Netzwerke mit wenigen Molekülen stochastisch modelliert werden können. Konkret werden wir dazu einen diskreten Zustandsraumes und die *stochastische Massenwirkungskinetik* benutzen.

Als erstes wird die Betrachtungsweise für die Angabe von Stoffmengen angepasst. Kontinuierliche Konzentrationen werden durch diskrete Molekülanzahlen ersetzt. Im weiteren Verlauf des Abschnittes soll ein biochemisches System mit u Stoffen $A_1, \ldots, A_u$ und w Reaktionen $R_1, \ldots, R_w$ betrachtet werden, dessen aktueller Zustand m, mithilfe der gegenwärtigen Anzahl der Moleküle aller Stoffe im Netzwerk, kodiert werden kann: $m := (\#A_1, \ldots, \#A_u) = (m[1], \ldots, m[u])$.

Es existieren noch weitere Interpretationen für diskrete Zustandskodierungen. Wir wollen uns aber zunächst auf die Deutung als Molekülanzahl konzentrieren.

Jeder Reaktion R_i ist eine *stochastische Ratenkonstante* c_i zugeordnet. Sie ist, wie ihr deterministisches Gegenstück k_i, ein fester Faktor.

Die Semantik des *stochastischen Massenwirkungsgesetzes* wird mittels der Ratenfunktion $h_i(m, c_i)$ für die Reaktion R_i formuliert. Bevor die physikalische Interpretation der Ratenfunktion $h_i(m, c_i)$ angegeben wird, soll die stochastische Bedeutung von $h_i(m, c_i)$ erläutert werden.

Angenommen das betrachtete System befindet sich zum Zeitpunkt t im Zustand m. Dann ist die Wahrscheinlichkeit, dass eine Reaktion R_i im nächsten infinitesimalen Zeit-

intervall $(t, t + dti]$ auftritt, gegeben durch den Wert $(h_i(m, c_i) \cdot dti)$.

Wir wollen hier einen ersten Zusammenhang zu den theoretischen Grundlagen aus Abschnitt (2.3) ziehen. Dort wurde ebenfalls in der Gleichung (2.21) die Wahrscheinlichkeit für eine Zustandswechsel, innerhalb eines infinitesimales Zeitintervalles, gebildet.

Bevor wir im Detail auf die Zusammenhänge zwischen stochastischer Massenwirkungskinetik, stochastischen Prozessen und stochastischen Petrinetzen eingehen, sollen abermals einführende Definitionen vorangestellt werden.

Definition 3.4 (Elementare stochastische Massenwirkungsreaktion)
Angenommen ein biochemisches Netzwerk enthält u Stoffe und kann in w unidirektionale Elementarreaktionen zerlegt werden. So kann eine einzelne Elementarreaktion beschrieben werden als

$$R_j : \qquad \sum_{i=1}^{u} l_{(i,j)} \cdot A_i \xrightarrow{c_j} \sum_{i=1}^{u} r_{(i,j)} \cdot A_i, \qquad j = 1, \ldots, w, \tag{3.8}$$

wobei A_i der i-te chemische Stoff und R_j die j-te elementare Reaktion ist. Die Faktoren $l_{i,j}$ und $r_{i,j}$ sind die stöchiometrischen Koeffizienten. Die Konstante c_j ist die stochastische Ratenkonstante für R_j.

Beispiel 8 (Netzwerk 4) *Im weiteren Verlauf dieses Abschnittes betrachten wir eine stochastische Version des Beispielnetzwerkes 3. Wir werden die Struktur der Reaktionen übernehmen und die deterministischen Ratenkonstanten durch die stochastischen Ratenkonstanten c_1 :=0.1, c_2 :=0.2, c_3 :=0.3 und c_4 :=0.4 ersetzen.*

Um die physikalische Interpretationen der Ratenfunktionen zu verdeutlichen, werden wir biochemische Reaktionen der ersten drei Ordnungen betrachten. Die Darstellungen der Ratenfunktionen gehen auf Gillespie [14] und Wilkinson [32] zurück.

Wenn in den folgenden Betrachtungen von Wahrscheinlichkeiten, für das Auftreten von Reaktionen gesprochen wird, handelt es sich jeweils um die Wahrscheinlichkeit $(h_i(m, c_i) \cdot dti)$, bezüglich des nächsten infinitesimalen Zeitintervalles dti (siehe Einleitung dieses Abschnittes). Die Punkte " ..." in den Darstellungen der Reaktionsgleichungen sollen den Umstand ausdrücken, dass die Interpretationen der Ratenfunktionen unabhängig von der Struktur des Reaktionsproduktes sind.

Ratenfunktionen für Reaktionen der 0. Ordnung

Angenommen eine biochemische Reaktion der 0. Ordnung hat die folgende Form.

$$R_i : \qquad \emptyset \xrightarrow{c_i} \ldots$$

Die stochastische Ratenkonstante c_i quantifiziert die Wahrscheinlichkeit, dass die Reaktion R_i auftritt.

$$h_i(m, c_i) := c_i.$$

Ratenfunktionen für Reaktionen der 1. Ordnung

Wir wollen eine biochemische Reaktion der 1. Ordnung mit der Form

$$R_i : \qquad A_j \xrightarrow{c_i} \ldots$$

betrachten. Die stochastische Ratenkonstante c_i repräsentiert die Wahrscheinlichkeit, dass ein einzelnes Molekül des Stoffes A_j in die Reaktion R_i eingeht. Theoretisch stehen $m[j]$ Teilchen von A_j für eine Reaktion zur Verfügung. Daraus ergibt sich eine kombinierte Gesamtwahrscheinlichkeit von

$$h_i(m, c_i) := c_i \cdot m[j].$$

Ratenfunktionen für Reaktionen der 2. Ordnung

Vorausgesetzt eine biochemische Reaktion der 2. Ordnung hat die folgende Form.

$$R_i : \qquad A_j + A_k \xrightarrow{c_i} \ldots$$

Die stochastische Ratenkonstante c_i bestimmt in diesem Fall die Wahrscheinlichkeit, dass ein einzelnes konkretes Paar von Molekülen der Stoffe A_j und A_k reagieren wird. Da in diesem Szenario $m[j]$ Moleküle des Stoffes A_j und $m[k]$ Moleküle von A_k vorausgesetzt werden können, ist die Anzahl verschiedener heterogener Paare gleich $(m[j] \cdot m[k])$. Man erhält dementsprechend eine kombinierte Gesamtwahrscheinlichkeit für diesen Reaktionstyp von

$$h_i(m, c_i) = c_i \cdot m[j] \cdot m[k].$$

Es existiert eine weitere Ausprägung der Reaktionen 2. Ordnung, die gesondert behandelt werden muss.

$$R_i : \qquad 2 \cdot A_j \xrightarrow{c_i} \ldots.$$

Abermals bestimmt die stochastische Ratenkonstante c_i die Wahrscheinlichkeit für das Reagieren eines einzelnen Paares. Diesmal handelt sich jedoch um ein homogenes Paar von Molekülen des gleichen Stoffes A_j. Es stehen deshalb lediglich $\frac{m[j]\cdot(m[j]-1)}{2}$ verschiedene Kombinationen von Molekülen des Stoffes A_j zur Reaktion bereit. Damit ergibt sich

$$h_i(m, c_i) = c_i \cdot \frac{m[j] \cdot (m[j] - 1)}{2} = c_i \cdot \binom{m[j]}{2}.$$

Die Gesetzmäßigkeiten dieses Interpretationsansatzes können direkt auf höhere Reaktionsordnungen und Reaktionen mit größeren stöchiometrischen Koeffizienten erweitert werden. Wir haben stets die stochastische Ratenkonstante c_i als festen Faktor. Diese stellt die Einzelwahrscheinlichkeit für das Reagieren einer minimalen notwendigen Menge von Ausgangsmolekülen dar. Der zweite Teil der Gleichung ist die die kombinatorische

Häufigkeit der minimal notwendigen Molekülmengen. Die beiden Teile werden zu der Gesamtwahrscheinlichkeit multipliziert, siehe Definition (3.9).

Wilkinson [32] weist darauf hin, dass Reaktionen höherer Ordnung (größer zwei) meist biochemische Effekte beschreiben, die aus Reaktionen der Ordnungen eins und zwei zusammen gesetzt sind. Diese sollten dementsprechend von vornherein als solche modelliert werden.

Beispiel 9 (Netzwerk 4) *Wir definieren die Ratenfunktionen $h_i(m, c_i)$ für unser Beispielnetzwerk 4, wobei $m := (\#A, \#B, \#C, \#D) = (m[A], m[B], m[C], m[D])$ die momentane Anzahl der Moleküle des Netzwerkes kodiert.*

$$
\begin{aligned}
h_1(m, c_1) &= c_1 \cdot \frac{m[A] \cdot (m[A] - 1)}{2} \cdot m[B] \\
h_2(m, c_2) &= c_2 \cdot m[C] \\
h_3(m, c_3) &= c_3 \cdot m[D] \\
h_4(m, c_4) &= c_4 \cdot m[C] \cdot \binom{m[D]}{3} = c_4 \cdot m[C] \cdot \frac{m[D] \cdot (m[D] - 1) \cdot (m[D] - 2)}{6}
\end{aligned}
$$

So ergibt sich zum Beispiel für $h_1((2, 3, 0, 2), 0.1)$ ein Wert von 0.3 und für $h_2((0, 1, 1, 2), 0.2)$ von 0.2.

3.3 Stochastische Petrinetze

Ein *stochastisches Petrinetz* ist ein bipartiter Graph, der aus Plätzen und Transitionen besteht. Der Zustand eines stochastischen Petrinetzes wird, mittels der aktuellen Markenanzahl auf den Plätzen, kodiert. Dieser Zustand verändert sich zeitlich geordnet, durch das zufallsgesteuerte Schalten von Transitionen. Die stochastischen Quantitäten des Modelles sind die Schaltraten, die einzeln für jede Transition definiert werden. Das Reziprok der Schaltrate gibt die mittlere verbleibende Zeit bis zum Schalten der Transition nach ihrer Aktivierung an.

Mittels der Netzstruktur, den spezifizierten Schaltraten und einer Anfangsbelegung der Plätze wird in eindeutiger weise ein stochastischer Prozess definiert. Eine Realisierung dieses Prozesses kann durch exakte oder approximative Simulationsverfahren berechnet werden.

Aufbauend auf dieser informalen Einführung, sollen im folgenden drei Definitionen von stochastischen Petrinetzklassen angegeben werden. Zwischen den zu definierenden Netzklassen besteht eine Untermengenbeziehung: $\mathcal{SPN} \subset \mathcal{GSPN} \subset \mathcal{DSPN}_{\mathcal{S}}$.

Definition 3.5 (Stochastisches Petrinetz)
Ein stochastisches Petrinetz ist ein Quintupel $SPN = (P, T, f, v, m_0)$, wobei

- *P und T sind endliche, nicht leere und disjunkte Mengen. Die Menge P enthält die Plätze und die Menge T die Transitionen des Netzwerkes.*

- *Die Relation* $f : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}_0$ *definiert die Menge der gerichteten Kanten, welche mit nicht-negativen ganzen Zahlen gewichtet sind.*
- *Die Funktion* $v : T \rightarrow H$ *ordnet jeder Transition* t *eine Ratenfunktion* h_t *zu, wobei* $H := \bigcup_{t \in T} \left\{ h_t \,|\, h_t : \mathbb{N}_0^{|{}^\bullet t|} \rightarrow \mathbb{R}^+ \right\}$ *die Menge aller Ratenfunktionen ist. Es gilt* $v(t) = h_t$ *für* $t \in T$.
- *Die Abbildung* $m_0 : P \rightarrow \mathbb{N}_0$ *gibt die Anfangsmarkierung an.*

Die Netzklasse der stochastischen Petrinetze soll durch $\mathcal{SPN}$ *symbolisiert werden.*

In der Definition der Netzklasse $\mathcal{SPN}$ werden die Begriffe der Markierung und die Menge ${}^\bullet t$ benutzt. Beide Elemente sollen im folgenden spezifiziert werden.

Definition 3.6 (Markierung und Zustandsraum eines $\mathcal{SPN}$)
Angenommen wird ein $SPN = (P, T, f, v, m_0) \in \mathcal{SPN}$. *Die Markierung eines Netzwerkes* $m : P \rightarrow \mathbb{N}_0$ *ist eine Funktion, welche die gegenwärtige Anzahl der Marken auf den verschiedenen Plätzen* $p \in P$ *festhält. Sie kann als* u*-dimensionaler Vektor betrachtet werden, der die Gestalt* $m = (\#token(p_1), \ldots, \#token(p_u))$ *besitzt. Die Menge* M *beinhaltet alle möglichen Markierungen, die das Netzwerk* SPN *einnehmen kann. Sie wird auch als Zustandsraum von* SPN *bezeichnet.*

Definition 3.7 (Menge der Vor- und Nachplätze einer Transition)
Angenommen wird ein $SPN = (P, T, f, v, m_0) \in \mathcal{SPN}$. *Die Menge der Vorplätze einer Transition* $t \in T$ *ist definiert als*

$$ {}^\bullet t := \{p \in P | f(p, t) \neq 0\}, \tag{3.9} $$

und die Menge der Nachplätze von t *entsprechend als*

$$ t^\bullet := \{p \in P | f(t, p) \neq 0\}. \tag{3.10} $$

Die Ratenfunktion h_t definiert im Allgemeinen die markierungsabhängige Schaltrate $\lambda_t(m)$ für eine Transition t. Der Wertebereich von h_t ist auf der Menge der Vorplätze ${}^\bullet t$ der Transition t eingeschränkt. Damit wird eine enge Verbindung von Netzstruktur und Schaltrate erzwungen.

Eine Transition muss, bis sie zum Schalten ausgewählt werden kann, zwei Aktivierungsstufen durchlaufen: die Aktivierung und die Konzessionierung. Sobald alle Vorplätze einer Transitionen ausreichend belegt sind, wird die Transition aktiviert.

Da in $\mathcal{SPN}$-Netzwerken alle Transitionen, bezüglich des Schaltens, gleichberechtigt sind, bedeutet die Aktivierung einer Transition, gleichzeitig ihre Konzessionierung zum Schalten. In Netzklassen mit verschiedenen Prioritätsstufen für Transitionen wird diese Äquivalenz von Aktivierung und Konzessionierung aufgehoben. Dort erhält eine aktivierte Transition erst die Konzession zum Schalten, wenn keine aktivierte Transition mit einer höheren Priorität vorliegt.

Wenn mehrere Transitionen gleichzeitig konzessioniert sind, wird diejenige Transition schalten, deren Wartezeit als erstes abgelaufen ist. Die eigentliche Veränderung der Markierung geschieht zeitlos und unterliegt der bekannten Schaltregel für diskrete Petrinetze

[1]. Nachdem die Markierung des Netzwerkes angepasst wurde, werden konzeptionell alle Wartezeiten neu berechnet und es findet wiederum ein Wettlauf um das nächste Schalten statt.

Diskrete Petrinetze modellieren das qualitative Verhalten eines betrachteten Systemes. In ihren Erreichbarkeitsgraphen ist das Systemverhalten aller möglichen Zeitabläufe abgebildet. Dieser Grundsatz bleibt auch für stochastische Petrinetze erhalten. Bei ihnen wird jedoch ein erwartetes zeitliches Verhalten durch stochastische Quantitäten hinzugefügt. Das erwartete zeitliche Verhalten bekommt inhärent eine wesentlich höhere Wahrscheinlichkeit zu geordnet, als ein stark abweichendes Verhalten. Prinzipiell bleibt jedoch der Erreichbarkeitsgraph in seiner Gesamtheit erhalten. Eine Konsequenz dieser Eigenschaft ist, dass die auf dem Zustandsraum basierenden Analysemöglichkeiten erhalten bleiben. Für eine genauere Betrachtung der Beziehungen zwischen diskreten, kontinuierlichen und stochastischen Petrinetzen verweisen wir auf [12].

Das Werkzeug für die Modellierung des quantitativen Systemverhaltens, ist die erwartete Zeitspanne zwischen Aktivierung und Schalten einer Transition. Die *Wartezeit bis zum Schalten* ist exponentiell verteilt und kann mit der Zufallsvariable RTF_t beschrieben werden.

Definition 3.8 (Wartezeit bis zum Schalten einer Transition)
Angenommen wird ein stochastisches Petrinetz $SPN = (P, T, f, v, m_0) \in \mathcal{SPN}$. Die Wartezeit bis zum Schalten einer Transition $t \in T$ ist eine exponentiell verteilte Zufallsvariable RTF_t. Die Dichtefunktion von RTF_t ist definiert als

$$f_{RTF_t}(\tau) = \lambda_t(m) \cdot \mathrm{e}^{(-\lambda_t(m)\cdot\tau)}, \qquad \tau \geq 0.$$

Demnach kann die durchschnittliche Wartezeit für die Transition t als der Erwartungswert $E[RTF_t] = \frac{1}{\lambda_t(m)}$ angegeben werden. Der Parameter $\lambda_t(m)$ drückt die Rate aus, wie oft die Transition t im Mittel in einer Zeiteinheit schaltet.

Aufgrund der exponentiell verteilten Wartezeiten RTF_t für alle Transitionen kann gefolgert werden, dass auch die Aufenthaltsdauern in den verschiedenen Markierungen exponentiell verteilte Zufallsgrößen sind, siehe Abschnitt (2.3.2). Bedenkt man zusätzlich den abzählbaren Zustandsraum M eines $\mathcal{SPN}$-Netzwerkes, ist die Äquivalenz zwischen dem stochastischen Prozess, der durch das $\mathcal{SPN}$-Netzwerk modelliert wurde, und einer kontinuierlichen Markow-Kette klar erkennbar.

Mittels folgender Regeln kann eine kontinuierliche Markow-Kette (Def. (2.11)) aus einem $\mathcal{SPN}$-Netzwerk abgeleitet werden [2].

- Der Zustandsraum der korrespondierenden CTMC entspricht dem Erreichbarkeitsgraphen des $\mathcal{SPN}$-Netzwerkes.
- Die Übergangsrate $q[k, l]$ von Zustand m_k zu m_l ist die Schaltrate $\lambda_t(m)$ der Transition t, welche von der Markierung m_k zur Markierung m_l führt.
- Falls mehrere Transitionen einen Übergang von der Markierung m_k zur Markierung m_l realisieren, ergibt sich die Übergangsrate $q[k, l]$ als Summe der Schaltraten $\lambda_t(m)$ der entsprechenden Transitionen.

Die Ratenfunktionen h_t, welche die Schaltraten $\lambda_t(m)$ bestimmen, können auf unser spezielles Anwendungsgebiet der biochemischen Netzwerke zugeschnitten werden.

Ratenfunktionen, bei denen die Netzmarken als einzelne Moleküle aufgefasst werden, wurden bereits im vorangegangenen Kapitel für stochastische Massenwirkungsreaktionen niedriger Ordnung vorgestellt. Wir geben an dieser Stelle eine verallgemeinerte Version als Definition an.

Definition 3.9 (Ratenfunktion für die Molekülinterpretation von Marken) *Angenommen wird ein $SPN = (P, T, f, v, m_0) \in \mathcal{SPN}$, das ein biochemisches Netzwerk mit einer stochastischen Massenwirkungskinetik modelliert. In dem Netzwerk werden die Marken als einzelne Moleküle interpretiert. Die Ratenfunktion $h(m, c_t) : M \times \mathbb{R}^+ \to \mathbb{R}^+$ für eine Transition $t \in T$ ist als*

$$h_t := c_t \cdot \prod_{p \in {}^\bullet t} \binom{m[p]}{f(p,t)}, \tag{3.11}$$

definiert. Der Faktor c_t ist die stochastische Ratenkonstante von t und $m[p]$ ist die aktuelle Anzahl der Marken auf dem Platz p.

Eine weitere Interpretation der diskreten Zustandskodierung wurde von Calder et. al in [3] vorgeschlagen. Die Marken werden bei diesem Modellierungsansatz als diskrete Konzentrationsintervalle betrachtet.

Man nimmt dabei an, dass die maximale molare Stoffkonzentration durch die Konstante MAX und die Anzahl der unterschiedlichen Intervalle durch $N + 1$ ausgedrückt wird. Dann repräsentieren die abstrakten Werte $0, \ldots, N$ die Konzentrationsintervalle 0, $(0,\ 1 * MAX/N]$, $(1 * MAX/N,\ 2 * MAX/N]$, $\ldots$, $(N - 1 * MAX/N,\ N * MAX/N]$. Jedes dieser endlich vielen diskreten Intervalle steht für eine Äquivalenzklasse von unendlich vielen kontinuierlichen Zuständen.

Definition 3.10 (Ratenfunktion für Intervallinterpretation von Marken) *Angenommen wird ein gewöhnliches $SPN = (P, T, f, v, m_0) \in \mathcal{SPN}$, das ein biochemisches Netzwerk mit einer stochastischen Massenwirkungskinetik modelliert. In dem Netzwerk werden die Marken als diskrete Konzentrationsintervalle interpretiert. Die Ratenfunktion $h(m, k_t) : M \times \mathbb{R}^+ \to \mathbb{R}^+$ für eine Transition $t \in T$ ist als*

$$h_t := k_t \cdot N \cdot \prod_{p \in {}^\bullet t} \left(\frac{m[p]}{N}\right), \tag{3.12}$$

definiert. Der Faktor k_t ist die deterministische Ratenkonstante, $m[p]$ ist die aktuelle Anzahl der Marken auf Platz p und N ist die Nummer des höchsten Intervalles.

3.4 Erweiterung um deterministische Transitionen

Im Laufe der Zeit wurden eine Vielzahl von verschiedenen Erweiterungen für die grundlegende stochastische Petrinetzklasse $\mathcal{SPN}$ vorgeschlagen, siehe [21] und [9]. Die wichtigste Modellierungserweiterung ist die Hinzunahme von *deterministischen Transitionen*.

Es gibt verschiedene Typen von deterministischen Transitionen. Generell wird hier die Wartezeit bis zum Schalten nicht mehr mit der zufälligen Realisierung einer Zufallsvariable ermittelt, sondern ist deterministisch, durch ein festes Zeitintervall vorgegeben. Zur Unterscheidung, von den neu eingeführten deterministischen Transitionen, wollen wir ab sofort die Transitionen mit probabilistisch erzeugter Wartezeit als *stochastische Transitionen* bezeichnen.

3.4.1 Generalisierte stochastische Petrinetze ($\mathcal{GSPN}$)

Generalisierte stochastische Petrinetze ($\mathcal{GSPN}$) werden aus der Klasse $\mathcal{SPN}$ abgeleitet. Dies geschieht durch die Hinzugabe von *Inhibitorkanten* und *unmittelbarer Transitionen* als Modellierungskomponenten. Die unmittelbaren Transition gehören zur Menge der deterministischen Transitionen.

Inhibitorkanten

Die Inhibitorkanten sind ein wesentliches Modellierungswerkzeug um die Berechnungsvollständigkeit zu erreichen. Damit besitzen die Petrinetze der Netzklasse $\mathcal{GSPN}$ die gleiche Ausdrucksstärke wie Turing-Maschinen [24].

Inhibitorkanten sind gerichtet und werden nur zwischen Plätzen und Transitionen gezogen. Führt eine Inhibitorkante zu einer Transition, ist diese nur aktiviert, wenn auf den entsprechenden Vorplätzen weniger Marken liegen, als das Inhibitorkantengewicht angibt.

Wenn ein Schaltvorgang statt findet, werden vom Vorplatz der Inhibitorkante keine Marken konsumiert.

Unmittelbare Transitionen

Die unmittelbaren Transitionen (Englisch: Immediate transitions) besitzen als deterministische Transitionsart keine probabilistisch verteilte Wartezeit. Für jede unmittelbare Transition ist die Wartezeit mit Null vereinbart. Sie schalten demnach sofort nach ihrer Konzessionierung. Eine weitere Besonderheit der unmittelbaren Transitionen ist die Tatsache, dass sie allen anderen deterministischen und stochastischen Transitionsarten priorisiert sind. Damit fällt für die unmittelbaren Transitionen die Aktivierung und die Konzessionierung zusammen.

In Situation, in denen mehrere unmittelbare Transitionen konzessioniert sind, wird bei einer Simulation, eine zufallsgesteuerte Auswahl getroffen [9]. In der Analyse werden jedoch alle möglichen Verhaltensweisen betrachtet.

Als Konsequenz kann man im Erreichbarkeitsgraphen zwischen *flüchtigen* und *konkreten* Zuständen unterscheiden. In flüchtigen Zuständen verbringt das System keine Zeit, bevor es in einen neuen Zustand wechselt. Dementsprechend ist die Aufenthaltsdauer in diesen Zuständen immer Null und nicht mehr exponentiell verteilt.

Als Semantik des Netzwerkes liegt somit keine kontinuierliche Markow-Kette mehr vor. Es besteht jedoch die Möglichkeit, die flüchtigen Zustände aus dem Erreichbarkeitsgraphen zu entfernen. Der reduzierte Erreichbarkeitsgraph entspricht dann wieder dem

Zustandsraum einer CTMC. Für eine genauere Beschreibung des Reduktionsverfahrens, verweisen wir auf [21].

Führt man Inhibitorkanten und unmittelbare Transitionen mit dem Grundformalismus der $\mathcal{SPN}$-Klasse zusammen, erhält man die weit verbreitete Klasse der generalisierten stochastischen Petrinetze $\mathcal{GSPN}$. Wir verzichten an dieser Stelle auf die Angabe einer formalen Definition und verweisen stattdessen auf [21].

3.4.2 Deterministische und stochastische Petrinetze ($\mathcal{DSPN}$)

Zeitbewertete Transitionen

Mit den unmittelbaren Transitionen haben wir bereits eine erste deterministische Quantifizierung für Wartezeiten kennen gelernt. Diese wurde mit dem Wert Null für alle unmittelbaren Transitionen fest vorgegeben. Diese Beschränkung wird mit der Einführung der *zeitbewerteten Transition* aufgehoben. Hier kann eine beliebige feste Wartezeit bis zum Schalten der Transition definiert werden.

Sobald nun eine zeitbewertete Transition t aktiviert ist, wird mittels der aktuellen Systemzeit und der festen Wartezeit für t ein Zeitpunkt berechnet, an dem der Schaltvorgang durchgeführt werden soll. Anders ausgedrückt, startet, nach der Aktivierung von t, ein Zeitmesser, der bis zur festen Wartezeit von t zählt. Verliert die entsprechende Transition vor dem berechneten Schaltzeitpunkt die Aktivierung, findet kein Schaltvorgang statt.

Betrachtet man Netzwerke ohne deterministische Transitionen, strebt die Wahrscheinlichkeit für das gleichzeitige Schalten zweier Transitionen gegen Null. Dagegen kann es in Netzwerken mit deterministischen Transitionen relativ oft vorkommen, dass Transitionen gleichzeitig schalten wollen. Ein einfaches Beispiel hierfür, ist die gleichzeitige Konzessionierung mehrerer unmittelbarer Transitionen. Abermals werden dann in der Analyse alle möglichen Verhaltensweisen betrachtet und in der Simulation findet eine zufallsgesteuerte Auswahl statt.

Definition 3.11 (Deterministisches und stochastisches Petrinetz in Snoopy)
Das 7-Tupel $DSPN_S = (P, T, f, g, v, l, m_0)$ ist ein deterministisches und stochastisches Petrinetz, wobei

- *P und T sind endliche, nicht leere und disjunkte Mengen. Die Menge P enthält die Plätze und die Menge T die Transitionen des Netzwerkes.*
- *Die Menge $T := (T_{stoch} \dot\cup T_{imm} \dot\cup T_{timed})$ ergibt sich als disjunkte Vereinigung der drei Transitionsmengen:*
  - *(i) T_{stoch}, welche stochastische Transitionen mit exponentiell verteilter Wartezeit enthält,*
  - *(ii) T_{imm}, welche unmittelbare Transitionen mit Wartezeit Null beinhaltet und*
  - *(iii) T_{timed}, welche zeitbewertete Transitionen mit einer deterministischer Wartezeit umfasst.*

- *Die Relation* $f : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}_0$ *definiert die Menge der gerichteten Kanten, welche mit nicht-negativen ganzen Zahlen gewichtet sind.*
- *Die Relation* $g : (P \times T) \rightarrow \mathbb{N}_0$ *definiert die Menge der gerichteten Inhibitorkanten, welche mit nicht-negativen ganzen Zahlen gewichtet sind.*
- *Die Funktion* $v : T_{stoch} \rightarrow H$ *ordnet jeder stochastischen Transition* $t \in T_{stoch}$ *eine Ratenfunktion* h_t *zu, wobei* $H := \bigcup_{t \in T_{stoch}} \left\{ h_t \,|\, h_t : \mathbb{N}_0^{|^\bullet t|} \rightarrow \mathbb{R}^+ \right\}$ *die Menge aller Ratenfunktionen und* $v(t) = h_t$ *für* $t \in T_{stoch}$ *ist.*
- *Die Funktion* $l : T_{timed} \rightarrow \mathbb{R}^+$ *spezifiziert für jede zeitbewertete Transition* $t \in T_{timed}$ *eine nicht-negative deterministische Wartezeit.*
- *Die Abbildung* $m_0 : P \rightarrow \mathbb{N}_0$ *gibt die Anfangsmarkierung an.*

Die Netzklasse der deterministischen und stochastischen Petrinetze in Snoopy soll durch $\mathcal{DSPN}_\mathcal{S}$ *symbolisiert werden.*

Die stochastischen Transitionen entsprechen den Transitionen der Netzklasse $\mathcal{SPN}$ und es gilt entsprechend die Definition (3.8) für deren Wartezeiten.

Die Netzklasse $\mathcal{DSPN}_\mathcal{S}$ ist eine Untermenge der Klasse $e\mathcal{DSPN}$, welche von German in [9] definiert wird. Um die Untermengenbeziehung deutlicher zu machen, werden folgende einschränkende Festlegungen für die Netzklasse $\mathcal{DSPN}_\mathcal{S}$, bezüglich der Eigenschaften von $e\mathcal{DSPN}$-Netzwerken heraus gestellt. Für eine Beschreibung der einzelnen Eigenschaften eines $e\mathcal{DSPN}$-Netzwerkes verweisen wir auf [9], Seite 19 folgend.

- Jeder stochastischen Transition $t \in T_{stoch}$ ist eine Exponentialverteilung mit dem Parameter $\lambda_t(m)$ für die Ermittlung der Wartezeiten zugeordnet,
- die Priorität der unmittelbaren Transitionen $t \in T_{imm}$ wird auf Null gesetzt,
- die Priorität der stochastischen und zeitbewerteten Transitionen $t \in (T_{stoch} \,\dot\cup\, T_{timed})$ wird auf Eins festgelegt,
- das Gewicht W für alle Transition $t \in T$ wird mit Eins definiert,
- die Markenabhängigkeit (*Markdep*) wird auf abhängig (*enabdep*) gesetzt,
- die Neuberechnungsstrategie der Wartezeiten (*policy*) wird auf gedächtnislos (*prd*) festgelegt,
- der Nebenläufigkeitsgrad (*concurrency*) wird mit schrittweise einzeln (*singleserver*) angegeben,
- es werden keine Prädikate (*Guards*) zur Steuerung der Aktivierung der Transitionen eingesetzt und
- es wird ebenfalls auf markenabhängige Kantengewichte (*marking-dependent arc multiplicities*) verzichtet.

Aufgrund der Tatsache, dass jedes $\mathcal{DSPN}_{\mathcal{S}}$-Netzwerk gleichzeitig ein $e\mathcal{DSPN}$-Netzwerk ist, kann die Theorie für die Modellierung und Analyse von $e\mathcal{DSPN}$-Netzwerken, wie sie in [9] und [16] entwickelt wird, angewendet werden.

Wir möchten bereits an dieser Stelle darauf hinweisen, dass das Werkzeug Snoopy zwei weitere Komponenten zur Modellierung anbietet: *Lesekanten* und *feste zeitbewertete Transitionen*. Diese Konzepte stellen jedoch nur eine vereinfachte Darstellung bereits definierter Bestandteile der Klasse $\mathcal{DSPN}_{\mathcal{S}}$ dar. Aus Gründen der Vergleichbarkeit mit der Klasse $e\mathcal{DSPN}$, werden sie in der Definition von $\mathcal{DSPN}_{\mathcal{S}}$-Netzwerken nicht als semantisch eigenständige Komponenten aufgeführt.

Lesekanten

Eine Lesekante erweitert prinzipiell die Modellierungsmächtigkeit nicht, sondern besitzt, bezüglich des Erreichbarkeitsgraphen, dieselbe Aussagekraft, wie zwei entgegengesetzt gerichtete und gleich gewichtete Kanten von einem Platz zu einer Transition. In einem Schaltvorgang werden somit genauso viele Marken konsumiert, wie produziert. Es findet dementsprechend auf dem Lesekantenvorplatz keine Veränderung der Markenanzahl statt. Vielmehr wird das Vorhandensein einer entsprechenden Anzahl von Marken als Schaltvorbedingung getestet bzw. gelesen. Im biochemischen Kontext kann, mittels der Lesekanten, komfortabel die Beteiligung von Enzymen an Reaktionen modelliert werden, da sich deren Molekülanzahl ebenfalls in der Regel nicht ändern sollte.

Feste zeitbewertete Transitionen

Die bisher betrachteten zeitbewerteten Transitionen besitzen einen relativen Zeitbezug. Erst nach ihrer Aktivierung wird ein eventueller Schaltzeitpunkt, relativ zur aktuellen Systemzeit, deterministisch berechnet.

Bei der Modellierung biochemischer Experimente besteht oft der Wunsch, bestimmte Impulse zu absoluten Zeitpunkten an das Netzwerk weiterzugeben. Das Schalten von Transitionen zu festen Zeitpunkten kann mit einer Kombination von zeitbewerteten und unmittelbaren Transitionen nachempfunden werden. Die Einführung von *festen zeitbewerteten Transitionen* ist dementsprechend keine Erweiterung der Modellierungsmächtigkeit.

Konstruktionsmuster für den Nachweis der Äquivalenz, zwischen zwei Varianten der festen zeitbewerteten Transitionen und einer Kombination von zeitbewerteten und unmittelbaren Transitionen, werden im Unterabschnitt (5.2.5) gegeben.

Kapitel 4

Simulation von stochastischen Petrinetzen

4.1 Simulation einer Zufallsvariable

Stochastische Petrinetze lassen sich analytisch nur schwer auswerten. Dies liegt unter anderem daran, das die Zustandsräume der zugrunde liegenden kontinuierlichen Markow-Ketten im Allgemeinen entweder unendlich sind oder zu groß für eine effiziente Analyse werden. Es wird deshalb auf eine computergestützte Simulation der Netzwerke zurück gegriffen. Dafür werden meist mehrere Simulationsläufe mit derselben Startkonfiguration berechnet. Anschließend kann anhand des Mittelwertes dieser Läufe ein charakteristisches Verhalten für das Netzwerk abgeleitet werden.

In diesem Kapitel wollen wir dem Leser zwei grundlegende Simulationsalgorithmen für stochastische Petrinetze aus der Klasse $\mathcal{SPN}$ vorstellen:

- die direkte Gillespie-Methode und
- die Poisson-Zeitschrittmethode.

Die beiden Algorithmen können jeweils als typische Vertreter für zwei grundsätzliche Klassen von Simulationsverfahren angesehen werden. Der erste Ansatz der *exakten Verfahren* führt eine schrittweise Simulation aller auftretenden Schaltereignisse durch. Das berechnete Ergebnis der Simulation entspricht damit zu jedem Zeitpunkt einer tatsächlichen Realisierung des stochastischen Prozesses, der mithilfe des $\mathcal{SPN}$-Netzwerkes modelliert wurde. Dagegen verzichten die Algorithmen der *approximativen Verfahren* auf die Berechnung jedes einzelnen Schaltvorganges. Sie fassen vielmehr eine Menge von Schaltvorgängen in einem einzelnen Berechnungsschritt zusammen. Jeder dieser Berechnungsschritte repräsentiert die geschätzte Netzveränderung, innerhalb eines bestimmten Zeitintervalles k. Die tatsächlich auftretenden Interaktionen, die innerhalb von k ablaufen, gehen nur als Annäherung in den Simulationsverlauf ein.

Die beiden Methoden werden im nächsten Abschnitt im Detail vorgestellt. Zunächst soll die Realisierung einer stetigen oder diskreten Zufallsvariable im Blickpunkt stehen. Das Erzeugen einer Zufallsvariablenrealisierung ist der wesentliche Grundbaustein einer

jeden stochastischen Simulation. Die Darstellung der Simulationsgrundlagen geht auf Wilkinson [32] zurück.

Egal, ob man die Realisierung einer stetigen oder diskreten Zufallsvariable betrachtet, es wird in beiden Fällen von einer gleich verteilten Zufallszahl $U \in [0,1]$ ausgegangen. Wir setzen für unsere Betrachtungen einen hinreichend guten Zufallszahlengenerator für das Intervall $[0,1]$ voraus.

Mithilfe der gleich verteilten Zufallszahl ist es möglich, die *Methode der inversen Verteilungsfunktion* für stetige Zufallsvariablen anzuwenden.

Proposition 4.1 (Methode der inversen Verteilungsfunktion)
Angenommen $\chi : \Omega \to \mathbb{R}$ ist eine stetige Zufallsvariable mit einer invertierbaren Verteilungsfunktion $F_\chi(\cdot)$. Sei $U \sim U([0,1])$ eine gleich verteilte Zufallszahl. Dann gilt

$$\chi := F_\chi^{-1}(U). \tag{4.1}$$

Beweis:

$$\begin{aligned} P(\chi \leq x) &= P(F_\chi(U) \leq x) \\ &= P(U \leq F_\chi(x)) \\ &= F_U(F_\chi(x)) \\ &= F_\chi(x) \qquad (wobei\ F_U(u) = u) \end{aligned}$$

□

Demnach ist es möglich, aus der Inversen F_χ^{-1} der Verteilungsfunktion, eine Realisierung von χ, mittels einer Instanz von $U \sim U([0,1])$ zu generieren.

Dieses Grundprinzip kann nun auf beliebige stetige Verteilungen mit einer invertierbaren Verteilungsfunktion angewendet werden. Eine essentielle Verteilung, für die Simulation der von uns betrachteten Netzklassen, ist die Exponentialverteilung. Eine Realisierung kann wie folgt erzeugt werden.

Proposition 4.2 (Simulation von $\chi \sim \mathbf{Exp}(\lambda)$)
Wenn $U \sim U([0,1])$ und $\lambda > 0$, dann besitzt die Zufallsvariable

$$\chi := -\frac{1}{\lambda} log(U) \tag{4.2}$$

eine exponentielle Verteilung mit dem Parameter λ.

Beweis: Angenommen $\chi \sim Exp(\lambda)$. Die Dichtefunktion $f_\chi(x)$ und die Verteilungsfunktion $F_\chi(x)$ von χ sind gegeben durch

$$f_\chi(x) = \lambda \cdot e^{-\lambda \cdot x}, \qquad F_\chi(x) = 1 - e^{-\lambda \cdot x}, \qquad x \geq 0.$$

Daraus folgt

$$F_\chi^{-1}(u) = -\frac{1}{\lambda}(1-u), \qquad 0 \leq u \leq 1.$$

Die Zufallszahlen U und $(1-U)$ besitzen beide die Verteilung $U([0,1])$. □

Die Poisson-Zeitschrittmethode basiert im wesentlichen auf der Realisierung von diskreten Poisson-verteilten Zufallsvariablen. Mittels einer Zufallsvariable $\chi_{poi} \sim Poi(\lambda)$ kann die Anzahl von Ereignissen simuliert werden, die innerhalb eines kleinen Zeitintervalles $\tau\iota$ auftreten. Die mittlere Häufigkeit der Ereignisse wird durch die Rate λ quantifiziert, die der Parameter der Poisson-Verteilung ist.

Für die Simulation von $\chi_{poi} \sim Poi(\lambda)$ wird die Eigenschaft ausgenutzt, dass die Zeitabstände $\tilde{\tau\iota}$ zwischen den Ereignissen exponentiell mit demselben Parameter λ verteilt sind. Es gilt somit $\tilde{\tau\iota} \sim Exp(\lambda)$. Es sei darauf hingewiesen, dass sich die Zeitintervalle $\tilde{\tau\iota}$ innerhalb des betrachteten Zeitintervalles $\tau\iota$ befinden, für das die Anzahl der Ereignisse simuliert werden soll.

Folgender einfacher Algorithmus kann demnach für die Realisierung von $\chi_{poi} \sim Poi(\lambda)$ aufgestellt werden:

(i) Initalisiere $\tau := 0$ und $number := 0$.

(ii) Simuliere $\tilde{\tau\iota} \sim Exp(\lambda)$.

(iii) Setze $\tau := \tau + \tilde{\tau\iota}$.

(iv) Falls $\tau < \tau\iota$, setze $number := number + 1$ und kehre zu Schritt (ii) zurück.

(v) result := number.

Das letzte benötigte Werkzeug für die beiden folgenden Simulationsalgorithmen, ist die Realisierung einer diskreten Zufallsvariable χ_d mit beliebiger Wahrscheinlichkeitsfunktion.

Das Grundprinzip für die Simulation von χ_d kann wie folgt beschrieben werden. Zunächst wird das Intervalles $[0,1]$ in einzelne Unterabschnitte zerlegt. Jeder Wert k den χ_d annehmen kann, bekommt genau einen dieser Abschnitte bijektiv zugeordnet. Die Größe des Unterintervalles k entspricht der Wahrscheinlichkeit $P(\chi_d = k)$.

Lässt man nun eine gleich verteilte Zufallszahl in das Intervall $[0,1]$ fallen, ergibt sich die Realisierung der diskreten Zufallszahl χ_d, als das entsprechende k des getroffenen Abschnittes.

Proposition 4.3 (Diskrete Bereichsmethode)
Angenommen $\chi_d : \Omega \to \mathbb{N}$ ist eine diskrete Zufallsvariable. Die Wahrscheinlichkeitsfunktion p_k von χ_d ist definiert als

$$p_k := P(\chi_d = k), \qquad für\ k = 0, 1, 2, \ldots.$$

Wir führen die spezielle Wahrscheinlichkeit q_k ein.

$$q_k := P(\chi_d \leq k) = \sum_{i=0}^{k} p_i.$$

Realisierungen von χ_d können erzeugt werden, indem man eine Zufallszahl $U \sim U([0,1])$ realisiert und anschließend

$$\chi_d := min\{k | q_k \geq U\}$$

berechnet.
Beweis: Die Wahrscheinlichkeit q_k ist genau so definiert, dass gilt $q_{k-1} < U \leq q_k$ für ein k. Daraus folgt

$$P(\chi_d = k) = P(U \in (q_{k-1}, q_k]) = q_k - q_{k-1} = p_k.$$

□

4.2 Die direkte Gillespie-Methode

Die direkte Gillespie-Methode wurde 1977 von Gillespie vorgeschlagen [14]. In der Literatur wird oft die allgemeinere Bezeichnung Gillespie-Algorithmus verwendet, obwohl Gillespie verschiedene Algorithmen zur Simulation von stochastischen Petrinetzen veröffentlicht hat [32].

Um die prinzipielle Arbeitsweise und den Hintergrund des Algorithmuses aufzuzeigen, werden wir zunächst einige Eigenschaften der kontinuierlichen Markow-Ketten (CTMC) wiederholen. Sie stellten die unterliegende Semantik eines $\mathcal{SPN}$-Netzwerkes dar, siehe Abschnitt (3.3).

Aus den Elementen der Übergangsmatrix $\mathbf{Q}$ einer CTMC konnten die Wahrscheinlichkeiten für einen Zustandsübergang, innerhalb des nächsten infinitesimalen Zeitintervalles dti, berechnet werden, siehe Gleichung (2.21). Wir wollen vereinbaren, dass wir uns im folgenden immer auf dieses nächste infinitesimale Zeitintervall dti beziehen, wenn wir von Wahrscheinlichkeiten im Zusammenhang mit Zustandswechseln sprechen.

Die negierten Diagonalelemente $-q[i,i]$ von $\mathbf{Q}$ repräsentierten die Wahrscheinlichkeit, dass sich das System aus dem Zustand s_i hinaus bewegt. Für die Diagonalelemente von $\mathbf{Q}$ galt $q[i,i] \leq 0$.

Die Aufenthaltsdauer in einem Zustand s_i, bis ein Zustandswechsel auftritt, konnte durch die Zufallsvariable SJ_{s_i} ausgedrückt werden. Dabei war SJ_{s_i} exponentiell mit dem Parameter $-q[i,i]$ verteilt, siehe Definition (2.16).

Die Nicht-Diagonalelemente $q[i,j]$ in der i-ten Zeile von $\mathbf{Q}$ gaben die Wahrscheinlichkeit an, dass ausgehend von Zustand s_i in einen Zustand s_j gewechselt wird. Das heißt, die Auswahl des nächsten Zustandes, den das System, verschieden von s_i, einnimmt, kann als eine diskrete Zufallsvariable $NT_i : \Omega \to (S - \{s_i\})$ betrachtet werden. Die Wahrscheinlichkeiten für die Werte aus $(S - \{s_i\})$ sind proportional zu den Werten der i-ten Zeile von $\mathbf{Q}$. Die folgenden zwei zentralen Aspekte können für die Simulation einer CTMC herausgestellt werden:

(i) die Simulation des Zeitintervalles, bis zum nächsten Zustandswechsel, als Realisierung der exponentiell verteilten Zufallsvariable SJ_{s_i} mit dem Parameter $-q[i,i]$ und

(ii) die Bestimmung des nächsten neuen Zustandes s_j, durch die Realisierung der diskreten Zufallsvariable NT_i mit den Wahrscheinlichkeiten $P(NT_i = s_j) = -\frac{q[i,j]}{q[i,i]}$ für $s_j \in (S - \{s_i\})$.

Wir betrachten ein Zeitintervall $[0, \tau_{max}]$. Der komplette Simulationsalgorithmus für ein $\mathcal{SPN}$-Netzwerk kann dann wie folgt angegeben werden:

(i) Initalisiere das $\mathcal{SPN}$-Netzwerk mit $\tau := 0$ und $m_{cur} := m_0$.

(ii) Ermittle die aktuelle Schaltrate $\lambda_t(m_{cur})$ für jede Transition $t \in T$, mittels der entsprechenden Ratenfunktion h_t.

(iii) Berechne die kombinierte Schaltrate $\lambda_{comb}(m_{cur}) := \sum_{t \in T} \lambda_t(m_{cur})$.

(iv) Simuliere das Zeitintervall $\tau\iota$ bis zum nächsten Schaltvorgang als exponentiell verteilte Zufallsvariable $SJ_{m_{cur}}$ mit dem Parameter $\lambda_{comb}(m_{cur})$.

(v) Setze $\tau := \tau + \tau\iota$.

(vi) Bestimme eine Transition $t_j \in T$, durch Simulieren der diskreten Zufallsvariable $NT_{m_{cur}}$. Die diskrete Zufallsvariable $NT_{m_{cur}}$ wird anhand der Wahrscheinlichkeiten $P(NT_{m_{cur}} = t) := -\frac{\lambda_t(m_{cur})}{\lambda_{comb}(m_{cur})}$ für alle $t \in T$ definiert.

(vii) Lasse die ausgewählte Transition t_j schalten und passe die aktuelle Markierung m_{cur} entsprechend an.

(viii) Gebe die veränderte Markierung m_{cur} und den aktuellen Zeitpunkt τ aus.

(ix) Falls $\tau < \tau_{max}$, kehre zu Schritt (ii) zurück.

Man kann aus der Arbeitsweise des Algorithmuses erkennen, dass es nicht notwendig ist, den gesamten Zustandsraum M des $\mathcal{SPN}$-Netzwerkes im Vorfeld zu kennen.

Falls das System unbekannte Zustände erreicht, wird in jeder Iteration die Übergangsratenmatrix $\mathbf{Q}$ (Def. (2.15)) um neue Elemente erweitert. Diese werden auf der Basis der aktuellen Schaltraten $\lambda_t(m_{cur})$ ermittelt. Die Berechnung der Schaltraten $\lambda_t(m_{cur})$ ist nur von der aktuellen Markierung m_{cur} abhängig, so dass keine unbekannten Zustände für die Simulation benötigt werden. Durch dieses schrittweise Vorgehen ist es möglich, dass auch Systeme mit einem sehr großen oder unendlichen Zustandsraum simuliert werden können.

Wir wollen daran erinnern, dass in einem $\mathcal{SPN}$-Netzwerk ein bestimmter Zustandsübergang nicht immer nur durch das Schalten genau einer Transition hervor gerufen werden kann. Die Übergangsrate $q[i, j]$ für einen solchen Zustandswechsel ergab sich aus der Summe der Schaltraten $\lambda_t(m)$ der entsprechenden Transitionen. Der Algorithmus trägt diesem Aspekt, in den Punkten (iii) und (vi) Rechnung. Punkt (iii) stellt eine Summe über alle Transitionen dar. Für Punkt (vi) gilt, dass die Auswahlvorgänge genau einer Transition $\{NT_{m_{cur}} = t_i\}$ jeweils disjunkte Ereignisse sind. Daraus folgt, dass die Wahrscheinlichkeit für die Auswahl einer Transition aus einer Menge von Transitionen, sich als die Summe der Einzelwahrscheinlichkeiten ergibt: $P(NT_{m_{cur}} \in \{t_1, \ldots, t_r\}) = \sum_{i=1}^{r} P(NT_{m_{cur}} = t_i)$. Die Menge $\{t_1, \ldots, t_r\}$ kann als Menge von Transitionen betrachtet werden, die ein und denselben Markierungswechsel initiieren.

Angepasste Version für die Klasse $\mathcal{DSPN}_S$

Für die Simulation von Netzwerken der Klasse $\mathcal{DSPN}_S$ (Def. (3.11)) muss der ursprüngliche Algorithmus, um die Behandlung deterministischer Transitionen, erweitert werden.

Wir rekapitulieren, dass die Transitionen in $\mathcal{DSPN}_S$-Netzwerken in stochastische, unmittelbare und zeitbewertete Transitionen unterteilt werden: $T := (T_{stoch} \dot\cup T_{imm} \dot\cup T_{timed})$.

Die Simulation von stochastischen Transitionen $t \in T_{imm}$ entspricht dem Vorgehen aus dem letzten Algorithmus.

Die unmittelbaren Transition $t \in T_{imm}$ können sofort nach der Änderung einer Markierung überprüft und gegebenenfalls ausgeführt werden.

Dagegen müssen alle aktivierten zeitbewerteten Transitionen zwischen gespeichert werden. Sie werden genau solange vorgehalten, bis sie entweder ihre Aktivierung verlieren oder ihr Zeitmesser abgelaufen ist. Im letzteren Fall schaltet die Transition, sobald sie die Konzession erhält. Dies geschieht nur dann, wenn keine aktivierten unmittelbaren Transitionen vorliegen. Trifft dies jedoch zu, werden die unmittelbaren Transitionen bevorzugt behandelt. Durch das priorisierte Schalten der unmittelbaren Transitionen, kann die Aktivierung der zeitbewerteten Transitionen wieder verloren gehen, bevor es zu einem Schaltvorgang kommt.

Wir führen für die Speicherung der aktivierten zeitbewerteten Transitionen die Menge $ATT \subseteq T_{timed}$ ein. Diese enthält alle gegenwärtig aktivierten zeitbewerteten Transitionen: ($\forall t \in T_{timed} : t \in ATT \Leftrightarrow$ t ist aktiviert). Für jede aktivierte Transition wird zusätzlich ihr jeweiliger absoluter Ablaufzeitpunkt $t.fire\_time$ fest gehalten. Dieser ergibt sich aus dem Zeitpunkt der Aktivierung, plus der deterministischen Wartezeit $l(t)$ (Def. (3.11)).

Wir fassen die Änderungsoperationen bezüglich der Menge ATT zu einem einfachen Teilalgorithmus $CheckATT(\tau, m_{cur})$ zusammen. Dieser bekommt die gegenwärtige Systemzeit τ und die momentane Markierung m_{cur} als aktuelle Eingabeparameter übergeben.

Teilalgorithmus $CheckATT(\tau, m_{cur})$:

(i) Teste, ob unaktivierte zeitbewertete Transitionen $t_i \in (T_{timed} - ATT)$ in der Markierung m_{cur} aktiviert wurden.

(ii) Wenn ja, füge entsprechende Transitionen t_i mit $t_i.fire\_time := \tau + l(t)$ in die Menge ATT ein.

(iii) Teste, ob aktivierte zeitbewertete Transitionen $t_j \in ATT$ in der Markierung m_{cur} ihre Aktivierung verloren haben.

(iv) Wenn ja, entferne entsprechende Transitionen t_j aus ATT.

Wir betrachten erneut das Simulationsintervall $[0, \tau_{max}]$. Eine angepasste Version der direkten Gillespie-Methode für die Netzklasse $\mathcal{DSPN}_S$-Netzwerk kann dann, wie folgt formuliert werden:

(i) Initalisiere das $\mathcal{DSPN}_S$-Netzwerk mit $\tau := 0$ und $m_{cur} := m_0$.

(ii) Solange eine oder mehrere unmittelbare Transitionen $t \in T_{imm}$ existieren, die in der Markierung m_{cur} konzessioniert sind, arbeite folgende Schritte ab:

- Wähle eine Transition t_j unter den konzessionierten unmittelbaren Transitionen gleich verteilt aus.
- Lasse die Transition t_j schalten und passe die aktuelle Markierung m_{cur} entsprechend an.
- Führe Teilalgorithmus $CheckATT(\tau, m_{cur})$ aus.

(iii) Wenn der Teilalgorithmus $CheckATT(\tau, m_{cur})$ in Punkt (ii) nicht mindestens einmal ausgeführt wurde, führe Teilalgorithmus $CheckATT(\tau, m_{cur})$ aus.

(iv) Ermittle die aktuelle Schaltrate $\lambda_t(m_{cur})$ für jede stochastische Transition $t \in T_{stoch}$, mittels der entsprechenden Ratenfunktion h_t.

(v) Berechne die kombinierte Schaltrate $\lambda_{comb}(m_{cur}) := \sum_{t \in T_{stoch}} \lambda_t(m_{cur})$ aller stochastischen Transitionen $t \in T_{stoch}$.

(vi) Simuliere das Zeitintervall $\tau\iota$ bis zum nächsten Schaltvorgang als exponentiell verteilte Zufallsvariable $SJ_{m_{cur}}$ mit dem Parameter $\lambda_{comb}(m_{cur})$.

(vii) Ermittle die Menge $ATT_{\tau_{min}} \subseteq ATT$ aller aktivierten zeitbewerteten Transitionen $t \in ATT$ mit minimaler Ablaufzeit $\tau_{min} := min\{t.fire\_time \mid t \in ATT\}$.

(viii) Wenn $(\tau_{min} < \tau + \tau\iota)$, arbeite folgende Schritte ab:

- Wähle gleich verteilt eine zeitbewertete Transition t_j aus der Menge $ATT_{\tau_{min}}$ aus.
- Entferne t_j aus ATT.
- Setze $\tau := \tau + \tau_{min}$.

(ix) Wenn $(\tau_{min} > \tau + \tau\iota)$, tue folgendes.

- Bestimme eine stochastische Transition $t_j \in T_{stoch}$, durch Simulieren der diskreten Zufallsvariable $NT_{m_{cur}}$. Die diskrete Zufallsvariable $NT_{m_{cur}}$ wird anhand der Wahrscheinlichkeiten $P(NT_{m_{cur}} = t) := -\frac{\lambda_t(m_{cur})}{\lambda_{comb}(m_{cur})}$ für alle $t \in T_{stoch}$ definiert.
- Setze $\tau := \tau + \tau\iota$.

(x) Lasse die Transition $t_j \in (T_{stoch} \dot{\cup} T_{timed})$ schalten, welche entweder in Punkt (viii) oder (ix) bestimmt wurde. Passe die aktuelle Markierung m_{cur} entsprechend an.

(xi) Die momentane Markierung m_{cur} und der dazugehörige Zeitpunkt τ wird ausgegeben.

(xii) Falls $\tau < \tau_{max}$, kehre zu Schritt (ii) zurück.

Die Wahrscheinlichkeit für den Fall $(\tau_{min} = \tau + \tau\iota)$ strebt gegen Null. Dementsprechend wird dieser Fall nicht behandelt.

4.3 Poisson-Zeitschrittmethode

Im letzten Abschnitt wurde ein exaktes Simulationsverfahren beschrieben. Es hat jedes einzelne Schalten einer Transition schrittweise nachgebildet. Jede Iteration repräsentierte das Schalten genau einer stochastischen oder zeitbewerteten Transition.

Wenn ein Netzwerk viele stochastische Transitionen mit sehr großen Schaltraten enthält, wird es dazu kommen, dass die Anzahl der Iterationen, innerhalb des Algorithmuses, stark ansteigt. Approximative Ansätze versuchen die Anzahl der Iterationsdurchläufe zu reduzieren, indem sie einen Genauigkeitsverlustes in der Abbildung des eigentlich modellierten stochastischen Prozesses in Kauf nehmen.

Hierfür wird das zu simulierende Zeitintervall $[0, \tau_{max}]$ in kleine Unterzeitabschnitte aufgeteilt, die jeweils genau einem Iterationsschritt im Algorithmus entsprechen. Die zentrale Annahme der Poisson-Zeitschrittmethode kann wie folgt formuliert werden. Die Größe $\tau\iota_k$ des jeweiligen Unterzeitabschnittes k wird so gewählt, dass die Schaltraten der Transitionen, innerhalb dieses Intervalles, als konstant angesehen werden können.

Wird die Prämisse als erfüllt vorausgesetzt, ist es nur noch notwendig, die Anzahl der auftretenden Schaltvorgänge für jede Transition $t \in T$, mittels einer Poisson-verteilten Zufallsvariable $TFN_t \sim Poi(\lambda_t(m_{cur}) \cdot \tau\iota_k)$ zu schätzen. Die Netto-Auswirkung aller dieser Schaltvorgänge ΔM_k kann unabhängig von einer konkret vorliegenden Markierung berechnet werden [1].

Wir setzen die Größen $\tau\iota_k$ der Unterzeitabschnitte als gegeben voraus und geben den Algorithmus zur Simulation eines $\mathcal{SPN}$-Netzwerkes an [32]:

(i) Initalisiere das $\mathcal{SPN}$-Netzwerk mit $\tau := 0$ und $m_{cur} := m_0$.

(ii) Ermittle die aktuelle Schaltrate $\lambda_t(m_{cur})$ für jede Transition $t \in T$, mittels der entsprechenden Ratenfunktion h_t.

(iii) Simuliere die Anzahl der Schaltvorgänge jeder Transition $t \in T$ als $TFN_t \sim Poi(\lambda_t(m_{cur}) \cdot \tau\iota_k)$ für den aktuellen Unterzeitabschnitt k.

(iv) Berechne die neue Markierung m_{cur} als Summe aus der alten Markierung und der Netto-Auswirkung aller geschätzten Schaltvorgänge ΔM_k. Die einzelnen Transitionen gehen mit den entsprechenden ermittelten Anzahlen TFN_t in ΔM_k ein.

(v) Setze $\tau := \tau + \tau\iota_k$.

(vi) Gebe die Markierung m_{cur} und die korrespondierende Zeit τ aus.

(vii) Falls $\tau < \tau_{max}$, kehre zu Schritt (ii) zurück.

Wir wollen den Punkt (iv) des Algorithmuses näher beleuchten. Die neue Markierung m_{cur} wird nicht mithilfe der Schaltregel ermittelt, sondern stellt vielmehr eine Summe aus alter Markierung und einer Schätzung für die Schaltvorgänge, innerhalb des Intervalles k, dar. Die kausalen Interaktionen werden dabei vernachlässigt, da die Vorbedingungen für die Schaltvorgänge, die in der Schätzung vorkommen, nicht berücksichtigt werden. Es

könnten somit Markierungen m_{cur} entstehen, die nicht im Zustandsraum des modellierten stochastischen Prozesses liegen.

Die zweite Vereinfachung gegenüber den exakten Verfahren, wurde mit der obigen Annahme bereits benannt. Innerhalb des Zeitabschnitts k werden die Schaltraten $\lambda_t(m_{cur})$ nicht neu berechnet, sondern werden als konstant vorausgesetzt.

Abschließend sei erwähnt, dass die Länge der Zeitabschnitte $\tau\iota_k$ dynamisch angepasst werden kann. Wilkinson gibt in [32] einen Überblick über entsprechende approximative Verfahren.

Kapitel 5

Implementierung

5.1 Analyse bestehender Programmpakete

In diesem Kapitel sollen etablierte Programme untersucht und vorgestellt werden, die sich im Umfeld der Modellierung und Simulation von stochastischen Petrinetzen und stochastischen biochemischen Netzwerken befinden.

Wir haben dazu jeweils einen typischen Vertreter aus den drei folgenden Bereichen gewählt:

- *TimeNET* als Modellierungswerkzeug für technische Netzwerke,
- *PIPE* als einen klassischen Petrinetz-Editor und
- *Dizzy* als Simulationswerkzeug für biochemische Netzwerke.

Folgende Gesichtspunkte stehen bei der Untersuchung im Mittelpunkt:

- Netzklassen
- grafische Editoren bzw. Eingabesprachen
- Animation und Simulation der Netzwerke
- Analyse der Netzwerke
- Im- / Export von Netzwerken

Alle betrachteten Werkzeuge sind jeweils unter den Betriebssystemen Microsoft Windows und Linux lauffähig.

5.1.1 TimeNET

TimeNET (Abb. (5.1)) ist ein ausgereiftes Programmpacket zur grafischen Modellierung und numerischen Analyse von stochastischen Petrinetzen. Bezieht man die Vorgängerversionen in die Betrachtung mit ein, wird es bereits seit 1991 an der Technische Universität Berlin entwickelt. In diesem Zeitraum hat TimeNET eine Vielzahl von Erweiterungen und Verbesserungen erfahren [33].

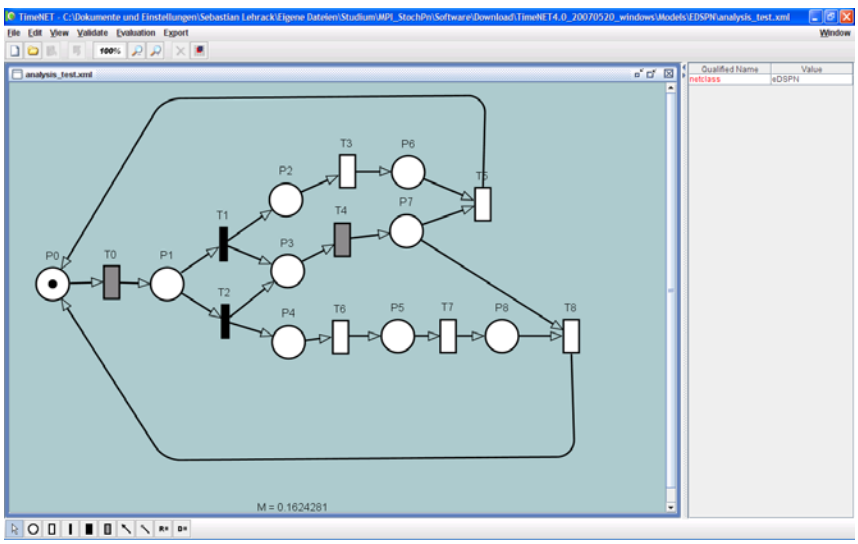

Abbildung 5.1: Editiermodus von TimeNET 4.0.

Netzklassen

Die aktuelle Version TimeNET 4.0 unterstützt die Netzklassen der *erweiterten determininistischen und stochastischen Petrinetze* ($e\mathcal{DSPN}$) und der *stochastischen gefärbten Petrinetze* ($\mathcal{SCPN}$), wie sie in [9] definiert werden.

In der Netzklasse $e\mathcal{DSPN}$ werden folgende verschiedene Klassen subsumiert.

- Generalisierte stochastische Petrinetze ($\mathcal{GSPN}$), siehe Unterabschnitt (3.4.1).
- Deterministische und stochastische Petrinetze ($\mathcal{DSPN}$), siehe Unterabschnitt (3.4.2).
- Erweiterte determin sind beliebige Wahrscheinlichkeitsverteilungen für Wartezeiten einer stochastischen Transition möglich.

Die Netzklasse der gefärbten stochastischen Petrinetze ($\mathcal{SCPN}$) erlaubt ebenfalls beliebige Wahrscheinlichkeitsverteilungen für die Wartezeiten einer stochastischen Transition. Weitere Charakteristika sind komplexe Marken, globale und zeitliche Prädikate und markenabhängige Transitionsprioritäten [33].

Grafische Editoren bzw. Eingabesprachen

TimeNET bietet einen Java-basierten grafischen Editor zur einfachen Modellierung der Netzwerkstrukturen an. Ein Mechanismus zur Erstellung hierarchische Netze oder logischer Knoten, siehe Anhang (A.1.1), ist nicht vorhanden.

Animation und Simulation der Netzwerke

Ein spezieller Animationsmodus für den Markenfluss wird nicht unterstützt. Für die beiden Netzklassen $e\mathcal{DSPN}$ und $\mathcal{SCPN}$ wurden Simulationsfunktionen integriert.

Analyse der Netzwerke

Mittels den vorhandenen Analysemöglichkeiten können strukturelle, sowie stochastische Eigenschaften ermittelt werden. So wird zum Beispiel die Berechnung einer stationären Verteilung unterstützt. Bemerkenswert ist die Option, dass der Editor und die Analysekomponenten auf verschiedenen Computern parallel zum Einsatz kommen können.

Im- / Export von Netzwerken

TimeNET arbeitet auf Grundlage eines proprietären XML-Formats. Im- und Exportfunktionen für Netzwerke stehen nur zum eigenen älteren TN-Format bereit. Simulationsergebnisse können im CSV-Format oder als SVG-Grafik (Simulationsplots) abgespeichert werden.

Fazit

Obwohl TimeNET im wesentlichen für die Modellierung und Analyse technischer Systeme und Protokolle entwickelt wurde, stellt es durch seine große Flexibilität und Funktionsvielfalt prinzipiell auch ein Werkzeug zur Modellierung biochemischer Netzwerke dar.

5.1.2 PIPE

Das Programm PIPE (Abb. (5.2)) ist ein klassischer Petrinetz-Editor, der auch strukturelle Analysekomponenten anbietet. Entstanden ist PIPE im Jahre 2003 innerhalb eines M.Sc.-Gruppenprojektes des Imperial College London. Obwohl es nach wie vor fortlaufend von dieser Forschungsgruppe weiterentwickelt wird, steht es mittlerweile auch als Open Source Projekt [18] zur Verfügung.

Netzklassen

Es wird explizit die Netzklasse der generalisierten stochastischen Petrinetze ($\mathcal{GSPN}$) unterstützt. Für die stochastischen Transitionen können in PIPE lediglich konstante Schaltraten $\lambda_t(m)$ definiert werden.

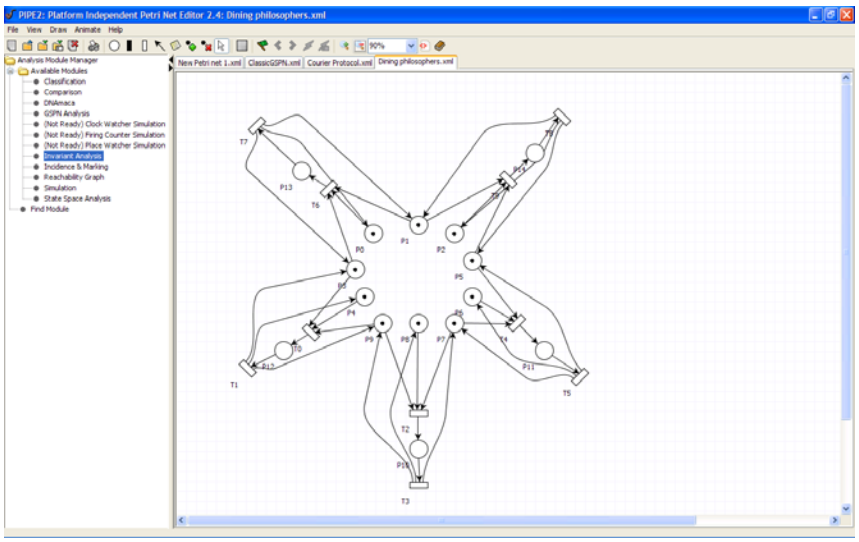

Abbildung 5.2: Editiermodus von PIPE.

Grafische Editoren bzw. Eingabesprachen

Ein Java-basierter Editor dient PIPE zur Eingabe der Netzwerkstrukturen. PIPE besitzt nicht die Fähigkeit zur Erstellung hierarchischer Netze oder logischer Knoten, siehe Anhang (A.1.1).

Animation und Simulation der Netzwerke

Der Markenfluss eines einzelnen Simulationslaufes kann in einem speziellen Modus animiert werden. PIPE bietet keine Algorithmen zur Simulation einzelner oder mehrerer Läufe an.

Analyse der Netzwerke

PIPE stellt eine Reihe von strukturellen und stochastischen Analysekomponenten für den Anwender bereit. So können unter anderem Invarianten [26], der Erreichbarkeitsgraph und die stationäre Verteilung des betrachteten Netzwerkes berechnet werden.

Im- / Export von Netzwerken

Die modellierten Netzwerkstrukturen werden mittels eines eigenen XML-Formates dauerhaft hinterlegt. Im- und Exportfunktionen zu anderen Petrinetzwerkzeugen werden nicht angeboten. Es besteht die Möglichkeit Netzwerkstrukturen als Grafik im PNG- oder PostScript-Format abzuspeichern.

Fazit

Ein Nachteil von PIPE ist das Nichtvorhandensein von Simulationsalgorithmen. Dadurch erscheint PIPE für den Einsatz im Kontext der Modellierung und Simulation von biochemischen Netzwerken nicht geeignet.

5.1.3 Dizzy

Dizzy (Abb. (5.3)) ist ein Programm, dass explizit für die kontinuierliche und stochastische Simulation von biochemischen Netzwerken entworfen wurde. Der maßgebliche Entwickler ist Stephen Ramsey vom Institut für Systembiologie in Seattle. [25]

Dizzy ist kein Programm das auf einer Petrinetzklasse basiert. Vielmehr wird hier auf den gleichen semantischen Modellen, wie bei vergleichbaren Petrinetzklassen aufgebaut. Zum einen sind das deterministische Differentialgleichungssysteme, wie bei kontinuierlichen Petrinetzen [28], und zum anderen Markow-Prozesse wie bei stochastischen Petrinetzen.

Bemerkenswert ist die Tatsache, dass eine Schnittstellen zur Programmsammlung *System Biology Workbench* (SBW) angeboten wird. Dizzy wird dementsprechend oft im Zusammenspiel mit weiteren Modellierungs- und Analyseprogrammen aus dieser Programmsammlung eingesetzt.

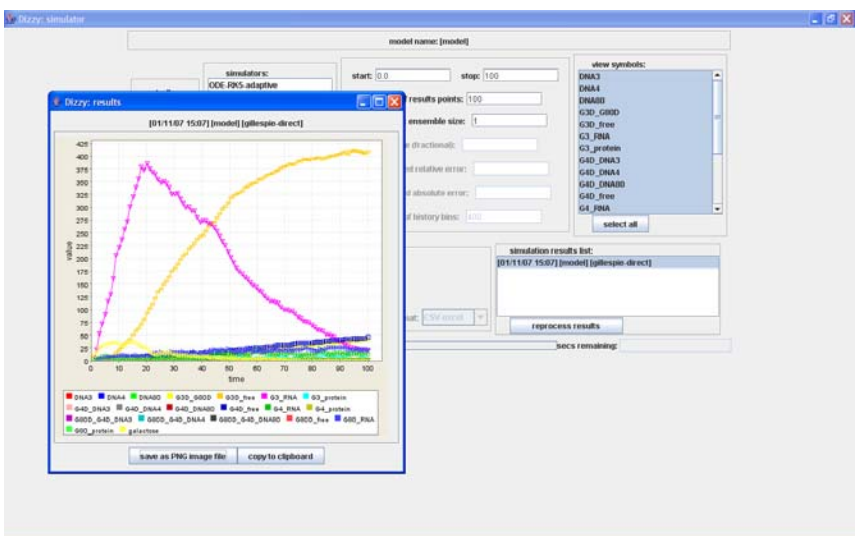

Abbildung 5.3: Simulationsergebnis als Plot in Dizzy.

Netzklassen

Da Dizzy keiner Petrinetzklasse unterliegt, sollen die wichtigsten Modellierungskomponenten von Dizzy und Snoopy gegenüber gestellt werden:

- Stoffe und Reaktionen in Dizzy sind äquivalent zu Plätzen und Transitionen.
- Es wird eine exponentiell verteilte Wartezeit für alle *normalen Dizzy-Reaktionen* [25] angesetzt.
- Zur Ermittlung der aktuellen Übergangsraten von normalen Dizzy-Reaktionen wird ein Äquivalent zur Ratenfunktion für die Molekülinterpretation von Marken (Def. (3.9)) eingesetzt.
- Es besteht die Möglichkeit, freie Funktionen für die Definition von Übergangsraten zu verwenden.
- Es können feste Wartezeiten für *verzögerte Dizzy-Reaktionen* [25] definiert werden. Dementsprechend sind auch unmittelbare Transitionen (ohne Priorisierung) abbildbar.

Grafische Editoren bzw. Eingabesprachen

Zur Eingabe der Modelle wird die Modellierungssprache CMDL (Chemical Model Definition Language) [25] oder das spezielle SBML-Format [17] verwendet. Beide Sprachen sind speziell für die Modellierung von biochemischen Netzwerken konzipiert worden und besitzen dementsprechend eine Vielzahl von Sprachkonstrukten zur schnellen Formulierung von Reaktionen. Als nachteilig ist das Fehlen eines grafischen Editors zu bezeichnen, der die Eingabe und das Verständnis der Netzstruktur verbessern würde.

Animation und Simulation der Netzwerke

Kennzeichnend für Dizzy ist das reichhaltige Angebot an Algorithmen für die deterministische und stochastische Simulation der erstellten Modelle.

Für eine deterministische Simulation stehen unter anderem Variationen des Runga-Kutta-Verfahrens und des Dormand-Prince-Verfahrens zur Verfügung. Sie stellen jeweils approximative numerische Lösungsverfahren für Differentialgleichungssysteme dar [28].

Zur stochastischen Simulation werden die direkte Gillespie-Methode (Abschnitt (4.2)), sowie das Gibson-Bruck-Verfahren [10] als Vertreter der exakten Verfahren angeboten, siehe Abschnitt (4.2). Daneben können Modelle auch mit dem approximativen τ-Leap-Algorithmus [32] simuliert werden.

Ein Animationsmodus ist aufgrund des Fehlens eines grafischen Editor nicht vorhanden.

Analyse der Netzwerke

Neben der Simulation stehen keine Komponenten für eine strukturelle oder stochastische Analyse der Modelle zur Verfügung.

Im- / Export von Netzwerken

Dizzy arbeitet wie bereits erwähnt auf der Basis der Sprachen CDML und SBML, die gleichzeitig weit verbreitete Austauschformate für Werkzeuge aus dem Umfeld der Systembiologie sind. Simulationsergebnisse können tabellarisch als CSV-Datei oder in Form eines Plots als PNG-Grafik exportiert werden.

Fazit

Dizzy ist durch die Umsetzung verschiedener Algorithmen für den Einsatz als Simulationswerkzeug prädestiniert. Negativ zu bemerken ist das Fehlen eines grafischen Editors zur Modellierung der Netzwerkstrukturen. Dizzy besitzt jedoch eine Schnittstelle zum Programmpacket SBW, mithilfe der Editoren angeschlossen werden können.

5.2 Leistungsmerkmale der neuen Petrinetzklasse in Snoopy

Wir wollen in diesem Abschnitt die Leistungsmerkmale von Snoopy, bezüglich der neu implementierten Netzklasse *Stochastic Petri Net* erläutern.

5.2.1 Strukturelle Komponenten

Die strukturellen Grundbausteine für ein stochastisches Petrinetz sind Plätze, Transitionen und gerichtete Kante zwischen jeweils einem Platz und einer Transition bzw. einer Transition und einem Platz.

Die Plätze und Transition werden in Snoopy, angelehnt an die diskreten Petrinetze, mit Kreisen und Rechtecken dargestellt. Dabei ist die grafische Ausprägung für alle drei Transitionsarten eines $\mathcal{DSPN}_{\mathcal{S}}$-Netzwerkes in der Snoopy-Grundeinstellung gleich. Eine manuelle Anpassung für jede einzelne Transition kann jederzeit durchgeführt werden [8]. So werden in den Beispielen der Diplomarbeit stochastische Transition als unausgefülltes Quadrat, eine zeitbewertete oder feste zeitbewertete Transition als ausgefülltes Quadrat und eine unmittelbare Transition als ausgefülltes schmales Rechteck dargestellt.

Die Kanten werden als durchgezogene schwarze Linie mit einem Pfeilende gezeichnet.

5.2.2 Zusätzliche Komponenten für die quantitative Modellierung

Neben den strukturellen Komponenten werden zusätzlich numerische Parameter und Wertetabellen zur Modellierung des Netzwerkes angeboten. Eine Wertetabelle stellt eine totale frei definierbare Funktion $f : \mathbb{R} \to \mathbb{R}$ dar. Beide Komponenten unterstützen die möglichst flexible Bildung von komplexen Ratenfunktionen.

5.2.3 Konfiguration für Animation und Simulation

Snoopy bietet die Möglichkeit, verschiedene Sätze bzw. Mengen für die nicht-strukturellen Einflussgrößen des Modells anzulegen. Es soll damit eine komfortable Anpassung dieser Größen für verschiedene Animations- und Simulationsanfragen gewährleistet werden.

Die verschiedenen Satztypen sind

- Anfangsmarkierungssätze,
- Raten- bzw. Semantikfunktionssätze und
- Parametersätze.

Es wäre also zum Beispiel denkbar, ein Netzwerk für verschiedene Kombinationen von Anfangsmarkierungen und Ratenfunktionen zu animieren bzw. zu simulieren.

5.2.4 Berechnung der Schaltraten

Die aktuellen Schaltraten $\lambda_t(m_{cur})$ werden durch die Ratenfunktionen h_t bestimmt, siehe Abschnitt (3.3). Schaltraten sind in der Regel markierungsabhängig und müssen daher bei einer auftretenden Markierungssänderung des Netzwerkes angepasst werden. Aus folgenden Bestandteilen kann die Ratenfunktion h_t einer Transition t zusammengesetzt sein:

- arithmetische Grundrechenoperationen,
- die aktuellen Markenanzahlen der Vorplätze von t,
- frei definierte Parameter,
- frei definierte Funktionen in Form einer Wertetabelle,
- vordefinierte Parameter und
- vordefinierte mathematische Funktionen.

Eine genaue Auflistung der vorgegebenen Operationen und Parameter wird im Anhang (A.1.6) gegeben.

BioMassAction(_stochasticRateConstant)

Snoopy bietet die Ratenfunktion für die Molekülinterpretation von Marken (Def. (3.9)) als vordefinierte Ratenfunktion an. Als Parameter von *BioMassAction(_stochasticRateConstant)* wird die stochastische Ratenkonstante c_i übergeben. Die eigentliche Struktur der Funktion wird intern, auf Grundlage des Netzaufbaues, gebildet.

BioLevelInterpretation(_deterministicRateConstant,_highestLevel)

Auch die Ratenfunktion für die Intervallinterpretation von Marken (Def. (3.10)) wird als vordefinierte Ratenfunktion angeboten. Als Parameter von *BioLevelInterpretation(_deterministicRateConstant,_highestLevel)* fungiert die deterministische Ratenkonstante k_i und die Nummer des höchsten Intervalles N. Die eigentliche Struktur der Funktion wird abermals intern, auf Grundlage des Netzaufbaues, gebildet.

5.2.5 Deterministische Transitionen

Wie in Abschnitt (3.4) diskutiert, wurde die Transitionsmenge der Klasse $\mathcal{DSPN}_S$ um deterministischen Transitionen erweitert. Für die Kennzeichnung von determintchen Transitionen werden spezielle *Semantikfunktionen* eingeführt. Sie definieren, anstelle der stochastischen Ratenfunktion, die Schaltsemantik für die entsprechende deterministische Transition. Wir werden anschließend die Abkürzungen RF und SF für Raten- und Semantikfunktion benutzen.

Es werden vier Semantikfunktionen angeboten:

(i) $ImmediateFiring()$,

(ii) $TimedFiring(\_relativeTime)$,

(iii) $FixedTimedFiring\_Single(\_absoluteTime)$ und

(iv) $FixedTimedFiring\_Periodic(\_intervalStart, \_timePeriod, \_intervalEnd)$.

Die Bedeutungen der Semantikfunktionen sollen im folgenden erläutert werden.

ImmediateFiring()

Transitionen mit der Semantikfunktion $ImmediateFiring()$ entsprechen den unmittelbaren Transitionen, wie sie in Unterabschnitt (3.4.1) eingeführt wurden.

TimedFiring(.)

Die Semantikfunktion $TimedFiring(\_relativeTime)$ kennzeichnet eine zeitbewertete Transition t, wie sie in Unterabschnitt (3.4.2) definiert wurde. Der Parameter $\_relativeTime$ bestimmt den Wert $l(t)$ (Def. (3.11)) für die zeitbewertete Transition t.

FixedTimedFiring_Single(.)

Die Semantikfunktion $FixedTimedFiring\_Single(\_absoluteTime)$ spezifiziert die erste Variante von festen zeitbewerteten Transitionen, siehe Unterabschnitt (3.4.2). Dieser Typ

versucht einen einzelnen Schaltversuch zum absoluten Zeitpunkt $\_absoluteTime$ durchzuführen. Wenn die Vorplätze der entsprechenden Transition, zum Zeitpunkt $\_absoluteTime$, nicht ausreichend markiert sind, findet kein Schaltvorgang statt.

Wir geben an dieser Stelle ein Konstruktionsmuster an, dass die bereits in Unterabschnitt (3.4.2) betonte Äquivalenz, zwischen festen zeitbewerteten Transitionen und einer Kombination von zeitbewerteten und unmittelbaren Transitionen, aufzeigen soll. Wir folgen den etablierten Konventionen und stellen eine stochastische Transition als unausgefülltes Quadrat, eine zeitbewertete oder feste zeitbewertete Transition als ausgefülltes Quadrat und eine unmittelbare Transition als ausgefülltes schmales Rechteck dar.

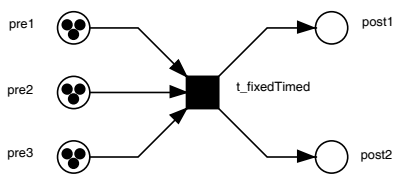

Abbildung 5.4: Ausgangsnetzwerk für Äquivalenznachweis.

In Abbildung (5.4) ist ein exemplarisches Ausgangsnetzwerk zu sehen. Es enthält lediglich eine feste zeitbewertete Transition $t\_fixedTimed$. Ohne Beschränkung der Allgemeinheit werden folgende Vereinbarungen für die Transition $t\_fixedTimed$ getroffen. Die Transition $t\_fixedTimed$ (SF: *FixedTimedFiring_Single(10)*) besitzt die drei Vorplätze *pre1*, *pre2* und *pre3* und die zwei Nachplätze *post1* und *post2*. Die Anfangsmarkierung ist mit m_0 := *(#pre1,#pre2,#pre3,#post1,#post2)* $= (3, 3, 3, 0, 0)$ gegeben und die Kantengewichte sind jeweils eins.

Dank, den ausreichend markierten Vorplätzen, schaltet $t\_fixedTimed$ nach exakt 10 Zeiteinheiten und es wird die Markierung $m_1 = (2, 2, 2, 1, 1)$ eingenommen.

Ziel ist es nun, dass von der Transition $t\_fixedTimed$ induzierte Netzverhalten, mittels einer Kombination von zeitbewerteten und unmittelbaren Transitionen, nach zu empfinden. Die Konstruktion des äquivalenten Netzwerkes kann in zwei zentrale Bestandteile unterteilt werden. Zum einen den Test, ob die Vorplätze der ursprünglichen Transition $t\_fixedTimed$ ausreichend belegt sind und zum zweiten, die Garantie, dass der Schaltversuch genau einmal nach exakt 10 Zeiteinheiten stattfindet. Das äquivalente Netzwerk ist in Abbildung (5.5) angegeben. Wir benutzen logische Knoten zur besseren Übersichtlichkeit, siehe Anhang (A.1.1).

Das konstruierte Netzwerk besteht aus

- den zwei Transitionen $switch\_test\_on$ und $switch\_test\_off$ zum Ein- und Ausschalten des korrekten Zeitpunktes,
- den zwei Transitionen $switch\_cond\_true$ und $switch\_cond\_false$ zum Auswerten der Schaltbedingung der ursprünglichen Transition $t\_fixedTimed$,
- der Transition $change\_post\_places$ zum Ändern der Markierung der Nachplätze von $t\_fixedTimed$ und

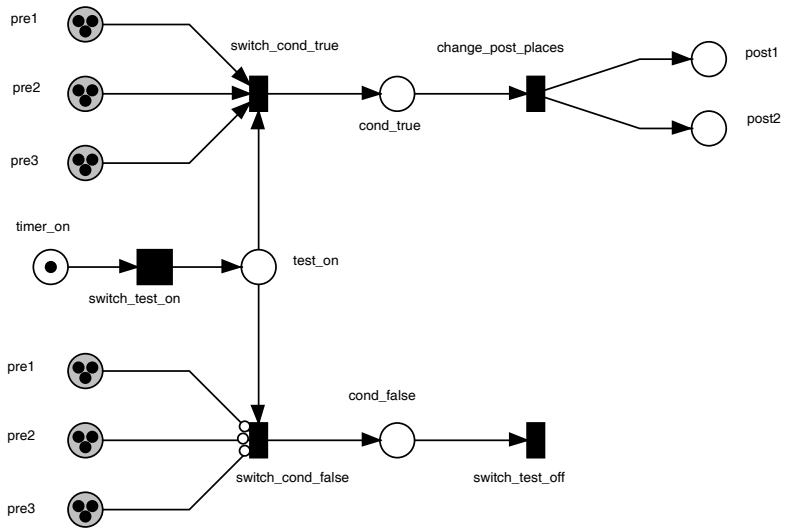

Abbildung 5.5: Äquivalentes Netzwerk für SF $FixedTimedFiring\_Single(.)$.

- den Plätzen $timer\_on, cond\_true$ und $cond\_false$.

Mit Ausnahme der zeitbewerteten Transition *switch_test_on* (SF: *TimedFiring(10)*), sind alle anderen Transitionen aus der Menge der unmittelbaren Transitionen (SF: *ImmediateFiring()*).

Die Auswertung der Schaltbedingung erfolgt mit den unmittelbaren Transitionen *switch_cond_true* und $switch\_cond\_false$. Genau eine von beiden schaltet, sobald der Platz *test_on* belegt ist. Erreicht wird das komplementäre Verhalten der Negativ-Auswertung mit dem Einsatz von Inhibitorkanten. Im negativen Auswertungsfall werden keine Marken konsumiert, da diese für die weitere Simulation des übrigen Netzwerkes weiter zur Verfügung stehen müssen.

Der Platz *test_on* wird genau einmal durch die Transition *switch_test_on* belegt, da der Vorplatz von *switch_test_on* nur eine Marke besitzt. Der Schaltvorgang findet genau nach 10 Zeiteinheiten statt, da die Transition *switch_test_on* bereits in der Anfangsmarkierung aktiv war und der Zeitmesser zu diesem Zeitpunkt abgelaufen ist.

Um die Korrektheit des Konstruktionsmusters zu bestätigen, müssen eventuell vorhandene Nebenbedingungen für die ursprüngliche Transition $t\_fixedTimed$ geprüft werden.

Teilen sich, ohne Beschränkung der Allgemeinheit, im Ursprungsnetz genau zwei feste zeitbewertete Transitionen $t\_fixedTimed_1$ und $t\_fixedTimed_2$, mit gleicher absoluter Schaltzeit, dieselben Vorplätze, würde eine gleich verteilte Auswahl zwischen diesen beiden Transitionen erfolgen. Dieser Vorgang wäre äquivalent mit dem gleichzeitigen Schaltversuch der beiden konstruierten Transitionen $switch\_test\_on_1$ und $switch\_test\_on_2$, da alle zeitbewerteten und festen zeitbewerteten Transitionen jeweils dieselbe Priorität besitzen.

Angenommen es haben, ohne Beschränkung der Allgemeinheit, eine feste zeitbewertete Transition $t\_fixedTimed$ und genau eine unmittelbare Transition t_imm die gleichen Vorplätze. Dann schaltet t_imm, gemäß der höheren Priorisierung, immer zuerst und kann somit der Transition $t\_fixedTimed$ die Aktivierung entziehen. Das äquivalente Verhalten

kann beobachtet werden, wenn die Transition *switch_test_on* und *t_imm* im konvertierten Netz zum gleichen Zeitpunkt aktiviert wären. Die Transition *switch_test_on* schaltet zwar nach dem Schaltvorgang von *t_imm*, jedoch würde die ursprüngliche Vorbedingung mittels $switch\_cond\_false$ gegebenenfalls negativ ausgewertet und die ursprüngliche Markierungsänderung, bezüglich der Transition $t\_fixedTimed$, bliebe auch hier aus.

FixedTimedFiring_Periodic(., ., .)

Die zweite Variante der festen zeitbewerteten Transitionen, mit der Semantikfunktion *FixedTimedFiring_Periodic(_intervalStart,_timePeriod,_intervalEnd)*, versucht im Zeitintervall [*_intervalStart*, *_intervalEnd*] periodisch einen Schaltvorgang durchzuführen. Mit den Parametern *_intervalStart* und *_intervalEnd* kann das Zeitintervall bezüglich der absoluten Zeitachse eingeschränkt werden. Der Rhythmus der Schaltversuche wird durch die Periode *_timePeriod* bestimmt.

Das äquivalente Konstruktionsmuster (Abb. (5.6)) für die periodische Version der festen zeitbewerteten Transitionen, entspricht im wesentlichen dem vorherigen Netzwerk. Es muss nur die Aktivierung der Transition *switch_test_on* angepasst werden. Anstatt mit einer einzelnen Marke, wird der Aktivierungsmechanismus nun, mithilfe der beiden festen zeitbewerteten Transitionen *switch_timer_on* und $switch\_timer\_off$, gebildet. Die beiden Transitionen besitzen die Semantikfunktionen *FixedTimedFiring_Single(_intervalStart)* und *FixedTimedFiring_Single(_intervalEnd)*. Die Transformation von *switch_timer_on* und *switch_timer_off* wurde bereits, mittels der ersten Variante, aufgezeigt.

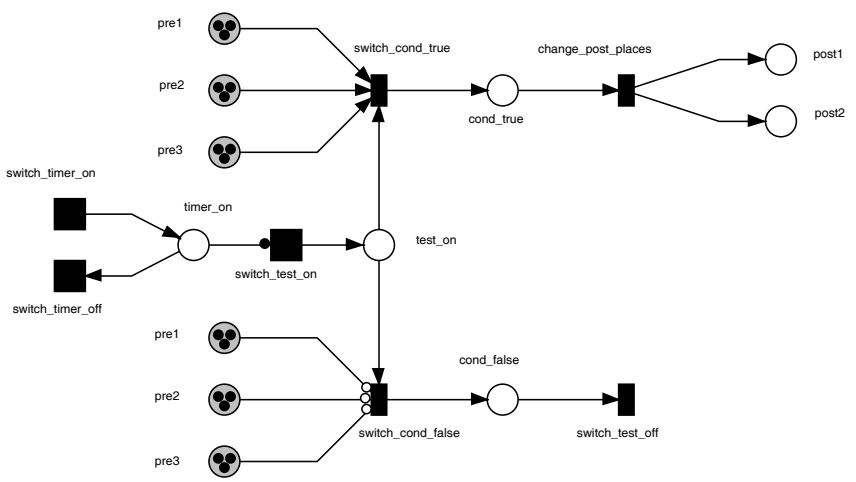

Abbildung 5.6: Äquivalentes Netzwerk für SF $FixedTimedFiring\_Periodic(.,.,.)$.

5.2.6 Animation des Netzwerkes

Für die Klasse der diskrete Petrinetze bietet Snoopy einen Animationsmodus an, in dem der Markenfluss nachvollzogen werden kann. Im Kontext der stochastischen Petrinetze ist der Markenfluss, die grafische Animation eines einzelnen Simulationslaufes.

Es findet jedoch nur eine angenäherte Wiedergabe des Simulationslaufes statt, da die Abstände zwischen den Schaltvorgängen konstant getaktet sind und die eigentlich zeitlosen Schaltvorgänge zur besseren Illustration animiert werden. Die Animation kann, mittels definierter Konfigurationssätze, parameterisiert werden.

5.2.7 Simulation des Netzwerkes

Snoopy simuliert das modellierte Netzwerk auf Grundlage der angepassten direkten Gillespie-Methode (Abschnitt (4.2). In einer einzelnen Simulationsanfrage wird eine vorgegebene Anzahl von Simulationsläufen berechnet. Die ermittelten Realisierungen sind voneinander unabhängig. Das Endergebnis der Simulationsanfrage ist die gemittelte Anzahl der Marken auf den Plätze zu bestimmten Ausgabezeitpunkten.

Eine Simulationsanfrage kann mit den folgenden Einstellungen parameterisiert werden:

- Konfigurationssätze für Markierungen, Ratenfunktionen und Parametern,
- Simulationsintervallende,
- Anzahl der Ausgabezeitintervalle und
- Anzahl der Simulationsläufe.

5.2.8 Darstellung der Simulationsergebnisse

Die Darstellung der Simulationsergebnisse geschieht in Snoopy entweder in Tabellenform oder als grafische Ausgabe in Form eines Plots. Für beide Ausgabearten können beliebig viele Ausprägungen erzeugt werden. So ist es zum Beispiel möglich, verschiedene Plots für unterschiedliche Platzmengen und Achsenauflösungen anzulegen.

Für eine Ergebnistabelle oder einen Plot kann die Auswahl der dargestellten Plätze eingeschränkt werden. Zusätzlich kann für einen Plot das dargestellte Zeit- und Wertachsenintervall angepasst werden.

5.2.9 Im- / Export

Grundsätzlich kann zwischen dem Im- und Export von Netzwerken, sowie dem Export von Simulationsergebnissen unterschieden werden.

Das Werkzeug Snoopy stellt generell eine Vielzahl von Exportmöglichkeiten zu weiteren Modellierungs- und Analyseprogrammen zur Verfügung. Für eine genaue Auflistung aller Programme, zu denen eine Exportfunktion existiert, wird auf den Anhang (A.4) verwiesen. Neben dem Export zu externen Programmen kann auch zwischen den verschieden Netzklassen von Snoopys konvertiert werden, siehe Anhang (A.4).

Für den Export von berechneten Simulationsergebnissen in das weit verbreitete CSV-Format, sind in Snoopy zwei prinzipielle Optionen vorgesehen. Zum einen wird ein automatisches Abspeichern jedes Ergebnisses einer Simulationsanfrage angeboten und zum anderem können Ergebnistabellen einzeln exportiert werden.

5.3 Validierung

Zur Validierung des implementierten Simulationsalgorithmuses (Abschnitt (4.2) wurde das Programmpacket *Discrete stochastic models test suite* (DSMTS) [6] benutzt. Es beinhaltet eine Reihe von einfachen Testnetzwerken, für die exakte, analytisch bestimmte, Systemverhalten vorliegen. Von einem korrekt arbeitenden Simulator wird erwartet, dass sich die gemittelten Simulationsergebnisse standardnormal verteilt um die analytischen Lösungen legen. Die jeweilig benutzte Standardnomalverteilung ist abhängig von der Anzahl der durchgeführten Simulationsläufe. Offensichtlich sollten die gemittelten Simulationsergebnisse, umso genauer mit der analytischen Lösung übereinstimmen, je mehr Läufe simuliert wurden. Für eine genauere Betrachtung, sei auf [6] verwiesen.

Der in Snoopy implementierte Algorithmus erfüllt, die in DSMTS aufgestellten Kriterien für einen korrekten Simulator.

Kapitel 6

Beispielanwendungen

In diesem Kapitel werden wir künstliche, sowie reale biochemische Netzwerke vorstellen, die mit dem Werkzeug Snoopy modelliert und simuliert wurden. Die Petrinetze der realen Modelle werden im Anhang (B.1 bis B.4) exakt definiert.

Wir folgen abermals den etablierten Konventionen und stellen eine stochastische Transition als unausgefülltes Quadrat, eine zeitbewertete oder feste zeitbewertete Transition als ausgefülltes Quadrat und eine unmittelbare Transition als ausgefülltes schmales Rechteck dar.

6.1 Konstruierte Ein- und Ausgabebausteine

In diesem Abschnitt sollen Modellierungsbausteine für die Steuerung von Zu- und Abflüssen von Stoffmengen vorgestellt werden. Wir werden auch hier die Akronyme RF und SF für die Begriffe Raten- und Semantikfunktion benutzen. Die Semantikfunktion aller unmittelbaren Transition ist per Definition als $ImmediateFiring()$ festgelegt. Sie wird daher im folgenden nicht explizit genannt.

Zeitgesteuerter Zu- und Abfluss

Im ersten einfachen Konstruktionsbeispiel (6.1) (Abb. (6.1)) betrachten wir eine einzelne reversible Reaktion $A \leftrightarrow B$. Sie wird durch die beiden Transitionen *t1* (RF: *BioMassAction(0.11)*) und *t2* (RF: *BioMassAction(0.1)*) modelliert. Die Reaktion $A \leftrightarrow B$ soll ein geschlossenes biochemisches System darstellen.

Der zeitpunktgenaue Zu- und Abfluss von Marken wird durch die beiden festen zeitbewerteten Transitionen *input* (SF: *FixedTimedFiring_Periodic(30,1,40)*) und *output* (SF: *FixedTimedFiring_Periodic(10,1,20)*) realisiert. Da an die Transition *input* keine Vorbedingungen gestellt werden, schaltet diese garantiert an den Zeitpunkten $11, 12, \ldots, 20$. Es werden dabei jeweils 1000 Marken zusätzlich auf den Platz A gelegt. Die Transition *output* dagegen, zieht zu den Zeitpunkten $31, 32, \ldots, 40$ jeweils 1000 Marken vom Platz B ab, falls zu den jeweiligen Zeitpunkten mindestens 1000 Marken vorhanden sind. Die Abbildung (6.2) zeigt einen einzelnen Simulationsverlauf für die ersten 100 Zeiteinheiten.

Als eine Variante (6.2) dieses Musters, wollen wir einen Abfluss aller aktuell vorhandenen Marken angeben (Abb. (6.3)).

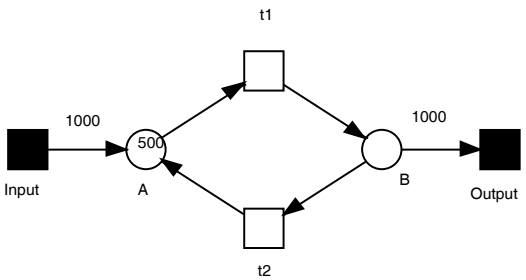

Abbildung 6.1: Beispielnetzwerk (6.1) für zeitgesteuerten Zu- und Abfluss.

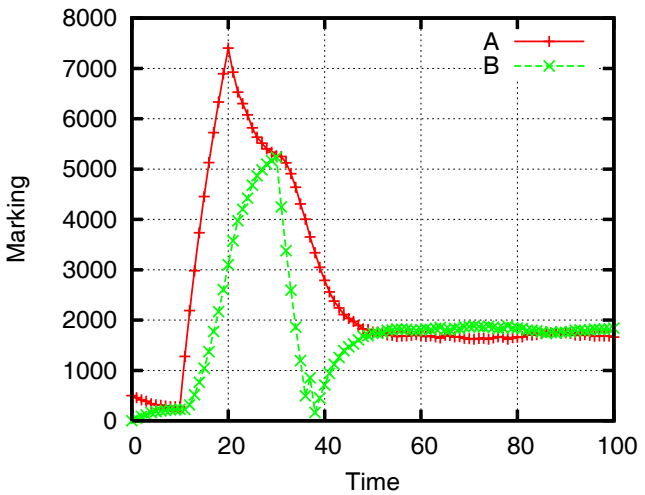

Abbildung 6.2: Simulationsergebnis für Beispielnetzwerk (6.1) (einzelner Lauf).

Das Beispielnetzwerk ist so konstruiert, dass in periodischen Zeitabständen alle Marken, die sich momentan auf Platz B befinden, aus dem System entfernt werden. Dafür konsumiert die unmittelbare Transition *output* alle Marken einzeln von Platz B, bis keine Marke mehr auf Platz *output_on* gelesen werden kann. Die Belegung von Platz *output_on* steuert die feste zeitbewertete Transition *switch_output_on* (SF: *FixedTimedFiring_Periodic(0,20,_SimEnd)*) und die unmittelbare Transition *switch_output_off*. Die feste zeitbewertete Transition *switch_output_on* initiiert, periodisch aller 20 Zeiteinheiten, einen Abfluss. Die unmittelbare Transition *switch_output_off* schaltet den Abfluss wieder ab, sobald alle Marken von B abgehoben wurden. In dem Fall, dass man den Abfluss nicht abschaltet, würde jede neu ankommende Marke auf Platz B unmittelbar entfernt. Ein einzelner Simulationslauf ist in Abbildung (6.4) angegeben.

Markengesteuerter Zufluss

Abermals wird eine reversible Reaktion $A \leftrightarrow B$ mit den stochastischen Transitionen *t1* (RF: *BioMassAction(0.1)*) und *t2* (RF: *BioMassAction(0.005)*) als geschlossenes System vorausgesetzt. In diesem Fallbeispiel (6.3), siehe Abbildung (6.5)), findet eine Erhöhung der Markenanzahl für Platz A um 50 statt, sobald die Markierung von Platz A unter 30 sinkt.

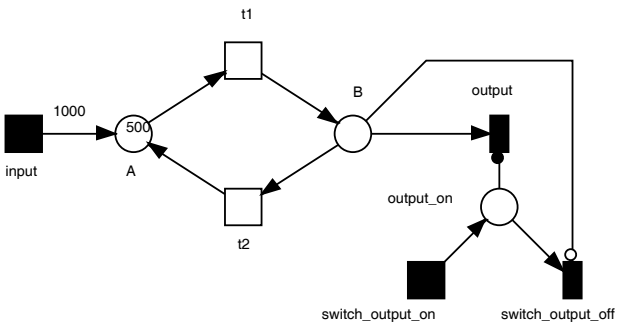

Abbildung 6.3: Beispielnetzwerk (6.2) für zeitgesteuerten Abfluss.

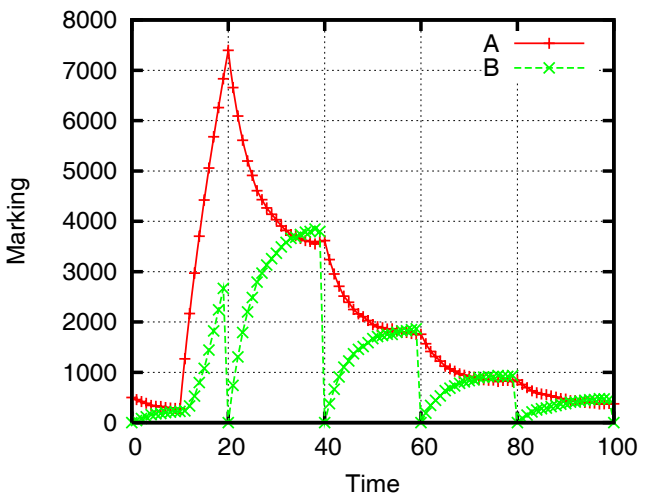

Abbildung 6.4: Simulationsergebnis für Beispielnetzwerk (6.2) (einzelner Lauf).

Realisiert wird dieses Verhalten durch die unmittelbare Transition *input* und die Inhibitorkante, welche von Platz A zur Transition *input* führt. Diese Inhibitorkante ist mit 30 gewichtet und verhindert somit das Schalten von *input*, genau solange bis die Markenanzahl des Platzes A unter 30 sinkt.

Ist der Sperre aufgehoben, werden sofort 50 Marken auf Platz A gelegt und ein weiterer Zufluss wird, bis auf weiteres, wieder verhindert. Die Grafik (6.6) zeigt das Simulationsverhalten eines einzelnen Laufes.

Ein zweites Beispielnetzwerk (6.4) für einen markengesteuerten Zufluss ist in der Abbildung (6.7) angegeben. Die Transitionen *t1* (RF: *BioMassAction(0.1)*) und *t2* (RF: *BioMassAction(0.2)*) bilden die reversible Reaktion $A \leftrightarrow B$.

Um ein signifikantes Verbrauchen von Marken für das System zu simulieren, ist die zeitbewertete Transition *output* (SF: *TimedFiring(5)*) integriert worden. Sie entfernt aller 5 Zeiteinheiten nach ihrer Aktivierung jeweils 10 Marken von Platz B. Eine Zufuhr von jeweils 5 Marken wird, mittels der zeitbewerteten Transition *input* (SF: *TimedFiring(0.5)*), immer genau dann initiiert, sobald die Anzahl der Marken auf Platz A unter 10 fällt. Der Zufluss um jeweils eine Marke auf Platz A geschieht periodisch aller 0.5 Zeiteinheiten nach der Aktivierung von *input*.

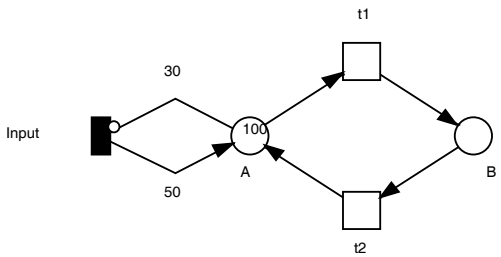

Abbildung 6.5: Beispielnetzwerk (6.3) für markengesteuerter Zufluss.

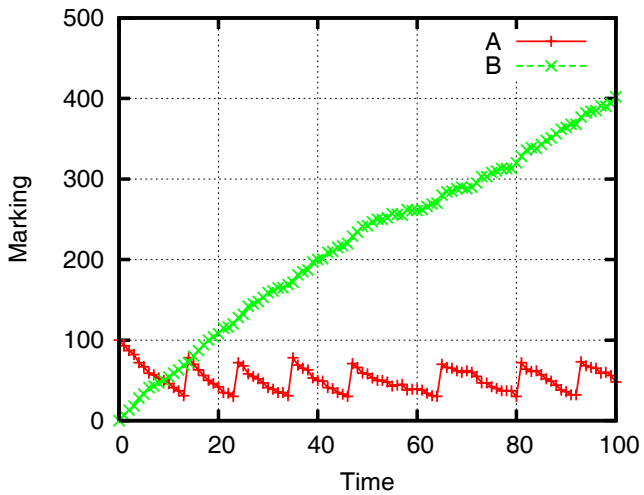

Abbildung 6.6: Simulationsergebnis für Beispielnetzwerk (6.3) (einzelner Lauf).

Die Steuerung, wann die Transition *input* schaltet, wird mittels den unmittelbaren Transitionen *switch_on* und *switch_off*, sowie den Plätzen *input_on* und $input\_off$ bewerkstelligt. Der Zufluss wird durch die Transition *switch_on* angeschaltet, sobald diese nicht länger mit mehr als 10 Marken auf Platz A gesperrt wird. Die einzelne Marke von Platz *input_on* wechselt, mittels der unmittelbaren Transition $switch\_off$, zu Platz $input\_off$, falls mindestens 30 Marken auf Platz A gelesen werden können.

Die beiden Grafiken der Abbildung (6.8) zeigen zum einen, einen isolierten Simulationslauf und zum anderen, das gemittelte Ergebnis von 100 Simulationsläufen. In der linken Grafiken kann man das An- und Abschalten (Platz *input_on*), bezüglich der Markierung von A, nachvollziehen. In der rechten Grafik dagegen, ist die Oszillation von Platz A und B zu erkennen, die auf das wechselnde An- und Ausschalten der Markenzufuhr basiert.

Wechsel von deterministischen zu stochastischen Transitionen

In den folgenden zwei Netzwerken soll exemplarisch ein einfaches Umschalten zwischen deterministischen und stochastischen Transitionen gezeigt werden. Zunächst wird ein zeitgesteuerter Wechsel vorgestellt, danach wird eine Möglichkeit aufgezeigt, wie ein markengesteuertes Umschalten realisiert werden kann.

Wir betrachten in beiden Fällen eine Reaktion $A \rightarrow B$, die durch die stochastische

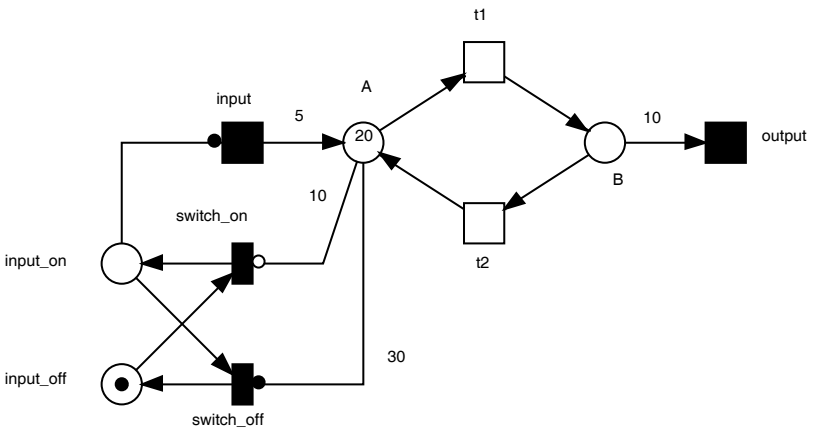

Abbildung 6.7: Beispielnetzwerk (6.4) für markengesteuerter Zufluss.

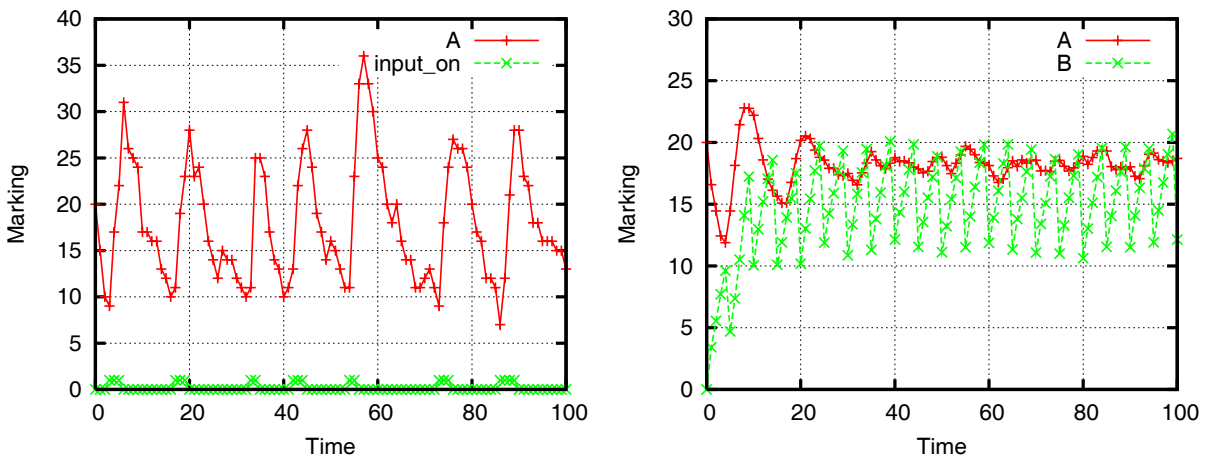

Abbildung 6.8: Ergebnisse für Beispielnetzwerk (6.4), Läufe: 1 (links) und 100 (rechts).

Transitionen *t_stoch* (RF: *BioMassAction(0.1)*) und die zeitbewertete Transition *t_det* (SF: *TimedFiring(0.25)*) modelliert wird. Durch die gewählte Netzstruktur ist zu jedem Zeitpunkt gewährleistet, dass nur eine der beiden Transitionen Marken von Platz A zu Platz B transferieren kann. Es findet also entweder ein stochastischer oder ein deterministischer Markenfluss statt.

Beim Beispiel (6.5) des zeitgesteuerten Umschaltens (Abb. (6.9)) wird eine einzelne Marke zwischen den beiden Plätzen *stochastic_on* und *det_on* bewegt. Dies geschieht durch die feste zeitbewertete Transitionen *switch_to_stoch* (SF: *FixedTimedFiring(10)*) und *switch_to_det* (SF: *FixedTimedFiring(10)*). Sie schalten zu den Zeitpunkten 10 und 30 zwischen den Transitionen *t_stoch* und *t_det* um. Diese sind mit ihren korrespondierenden Plätzen *stochastic_on* und *det_on*, mittels jeweils einer Lesekante verbunden.

Das Grundprinzip bleibt für das markengesteuerte Umschalten (6.6), siehe Abbildung (6.10), erhalten. Auch hier wird die aktuelle Information, ob die stochastische Transition *t_stoch* oder die deterministische Transition *t_det* aktiv ist, in den Plätzen *stochastic_on* und *det_on* kodiert.

Sobald die Markenanzahl auf Platz A unter die Grenze von 700 fällt, wird einmalig die

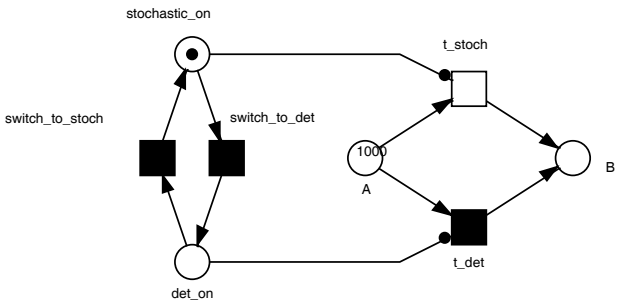

Abbildung 6.9: Beispielnetzwerk (6.5) für zeitgesteuertes Umschalten.

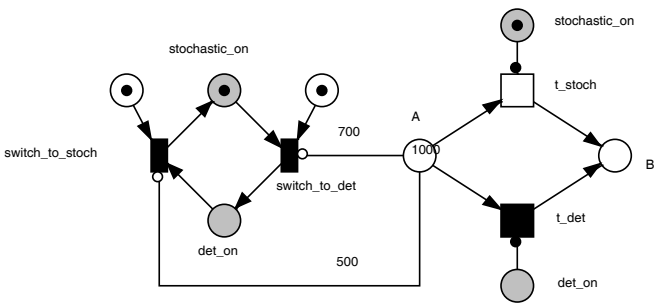

Abbildung 6.10: Beispielnetzwerk (6.6) für markengesteuertes Umschalten.

unmittelbare Transition *switch_to_det* geschaltet und die Transition *t_stoch* wird de- und die Transition *t_det* wird aktiviert. Die Reaktion $A \rightarrow B$ schaltet solange deterministisch, bis die Anzahl der Marken auf Platz A unter 600 sinkt. Tritt dieser Fall ein, schaltet die unmittelbare Transition *switch_to_stoch* und die Plätze *stoch_on* und *det_on* werden entsprechend belegt.

Die Grafiken der Abbildung (6.11) zeigen das jeweilige Verhalten der beiden Varianten für einen einzelnen Simulationslauf.

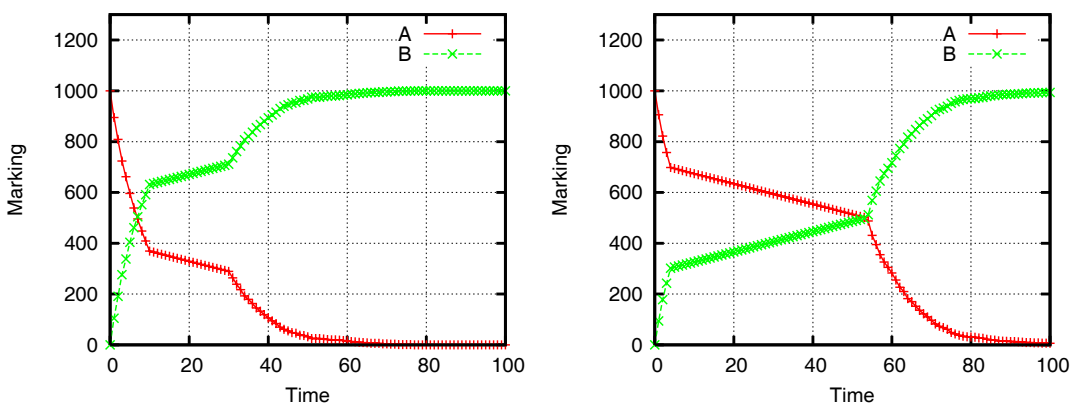

Abbildung 6.11: Ergebnisse für die Netzwerke (6.5) und (6.6) (einzelne Läufe).

6.2 Jäger-Beute-Modell von Lotka-Volterra

Das einfache Jäger-Beute-Modell von Lotka-Volterra (Abb. (6.12)) repräsentiert ein künstliches Ökosystem, in dem die Interaktion der Populationen von Pflanzen- und Fleischfressern modelliert wird. Bereits 1976 hat Gillespie [13], anhand dieses Beispiels, die Notwendigkeit einer stochastischen Betrachtungsweise gegenüber der deterministischen Modellierung aufgezeigt. Die beiden Grafiken in Abbildung (B.1) stellen die Ergebnisse einer stochastischen und einer deterministischen Simulation gegenüber.

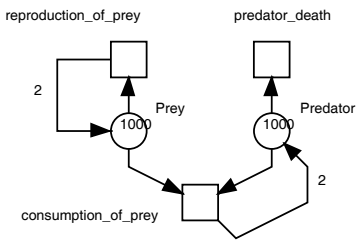

Abbildung 6.12: Jäger-Beute-Modell von Lotka-Volterra.

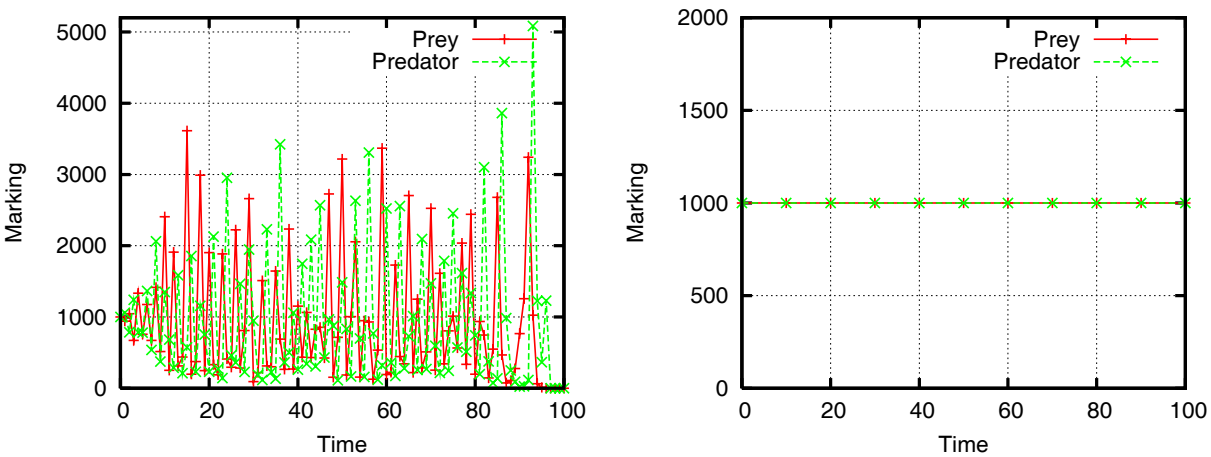

Abbildung 6.13: Simulation des Jäger-Beute-Modelles: stoch. (links), det. (rechts).

6.3 Zellzyklus-Modell von Tyson

Die Abbildung (6.14) zeigt das Zellzyklus-Modell von Tyson [30]. Dieses offene Modell besitzt die Transitionen r_1 und r_7 für den Zu- und Abfluss von Marken. Charakteristisch für dieses Modell ist die Oszillation der beteiligten Stoffe, siehe Abbildung (6.14) und Anhang (B.2). Auslöser dieser Oszillation ist die Autokatalyseeigenschaft der Transition r_4p [30].

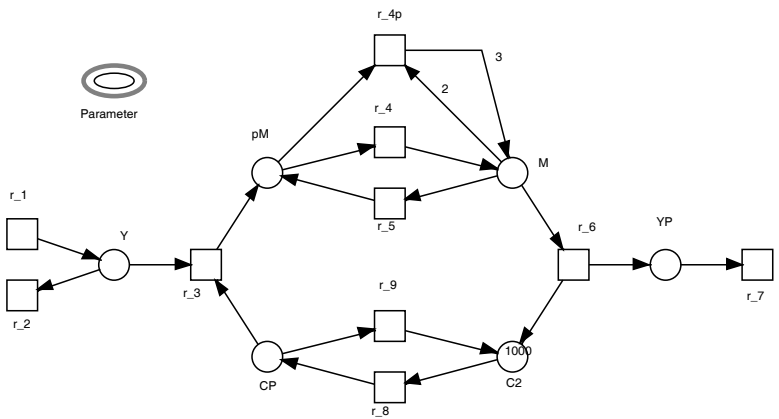

Abbildung 6.14: Zellzyklus-Modell von Tyson.

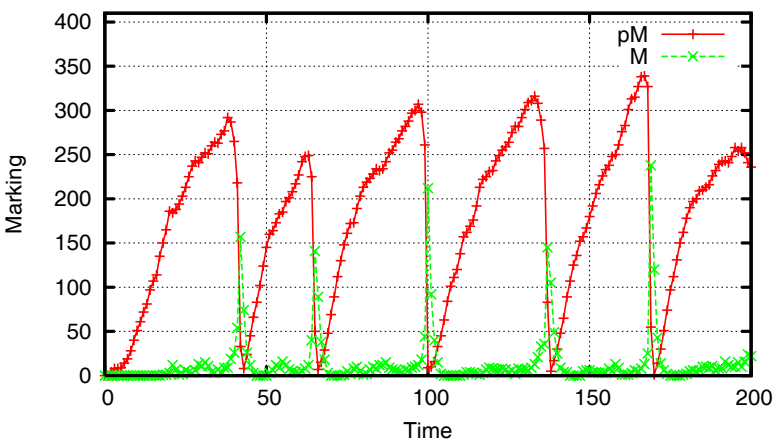

Abbildung 6.15: Simulationsergebnis vom Zellzyklus-Modell von Tyson: pM, M.

6.4 MAPK-Kaskade von Levchenko

Das hier betrachtete Modell der MAPK-Kaskade wurde als Differentialgleichungssystem in [20] von Levchenko vorgestellt. Dieses Netzwerk wurde unter anderem als Fallstudie in [12] ausführlich untersucht. Dort werden die Modellierungs- und Analysefähigkeiten von diskreten, stochastischen und kontinuierlichen Petrinetzen illustriert und in Beziehung gestellt. Wir geben hier eine Version als Beispiel für den Konzentrationsintervallansatz von Calder et. al an, siehe Abschnitt (3.3). Simulationsergebnisse und die komplette Definition des Modells ist im Anhang (B.3) zu finden.

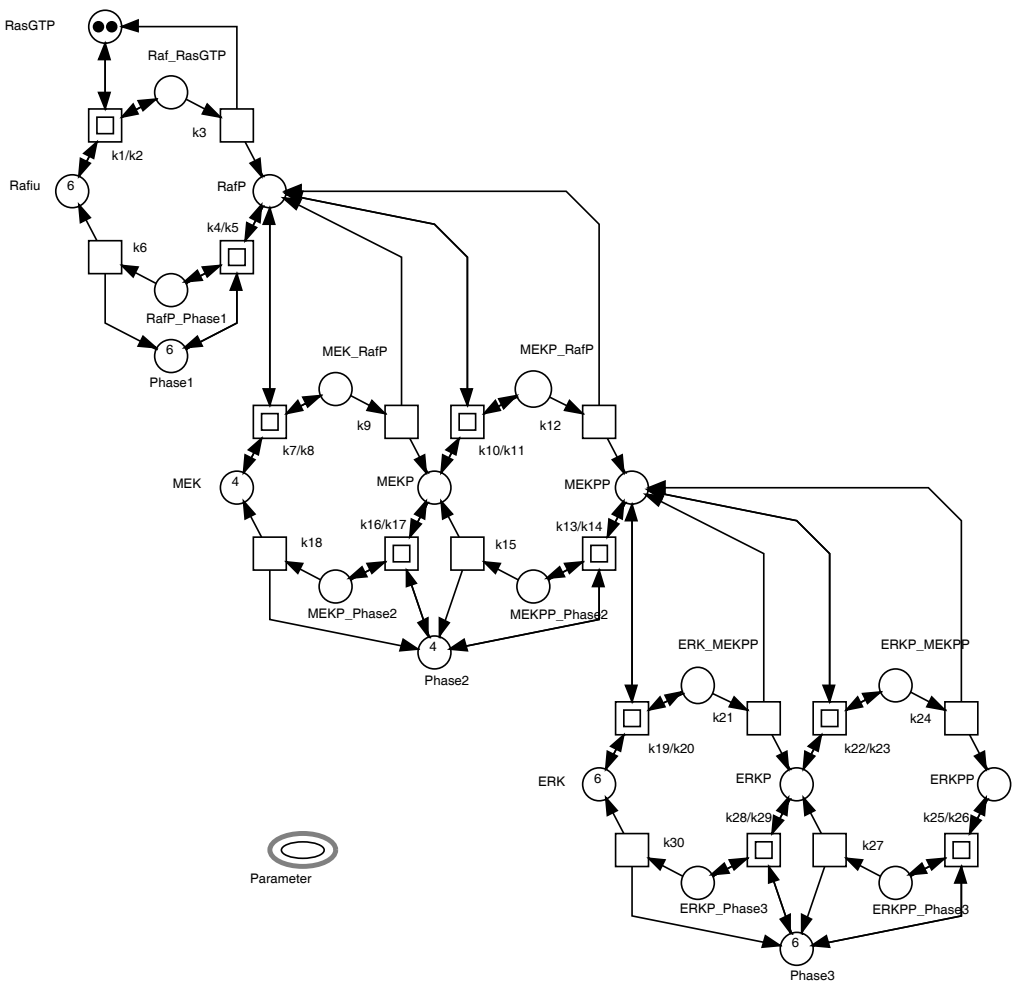

Abbildung 6.16: MAPK-Kaskade von Levchenko.

6.5 Lac Operon-Modell

Dieses Modell (Abb. (6.17) und Anhang (B.4)) wird von Wilkinson [32] als klassisches Beispiel eines Genregulationsnetzwerkes angegeben. Das Laktose-Operon spielt eine wichtige Rolle beim Transport und beim Abbau von Laktose in Bakterien.

An diesem Modell soll der Einsatz einer festen zeitbewerteten Transition gezeigt werden. Die Transition *Intervention* (SF: *FixedTimedFiring_Periodic(0,50000,_SimEnd)*) erhöht die Anzahl der Laktose-Moleküle, um jeweils 10000, aller 50000 Zeiteinheiten, siehe Abbildung (6.18). Der Platz Z symbolisiert das Enzym β-Galactosidase. In der Abbildung (6.19) kann die entsprechende Reaktion auf die Hinzugabe der Laktose-Moleküle nachvollzogen werden.

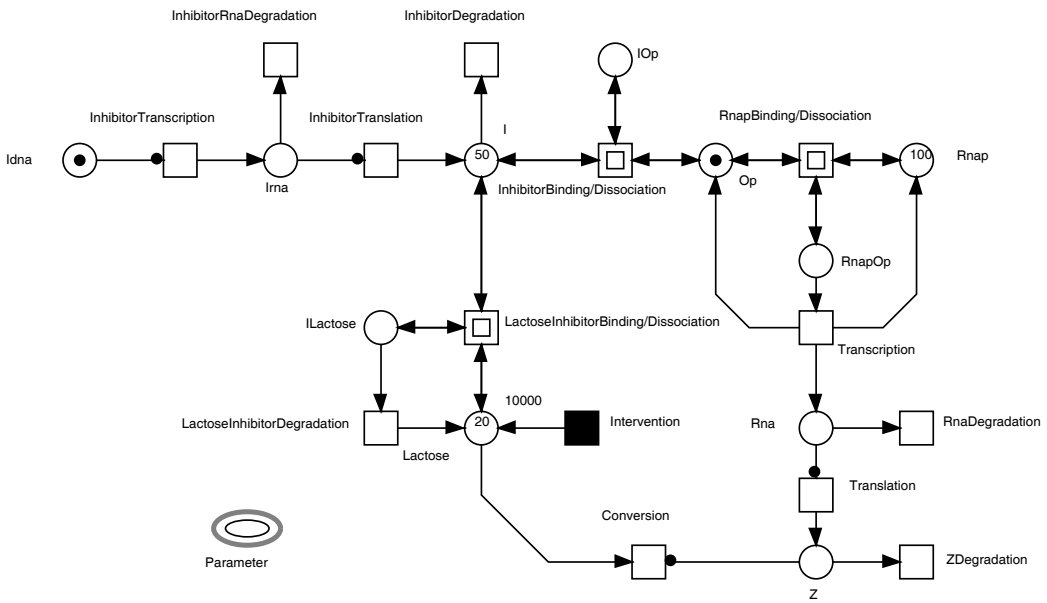

Abbildung 6.17: Lac Operon-Modell.

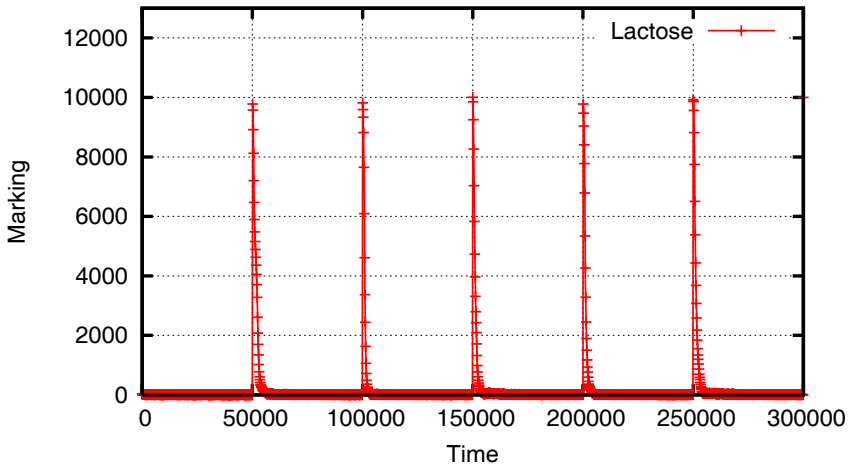

Abbildung 6.18: Simulationsergebnis vom Lac Operon-Modell: Laktose.

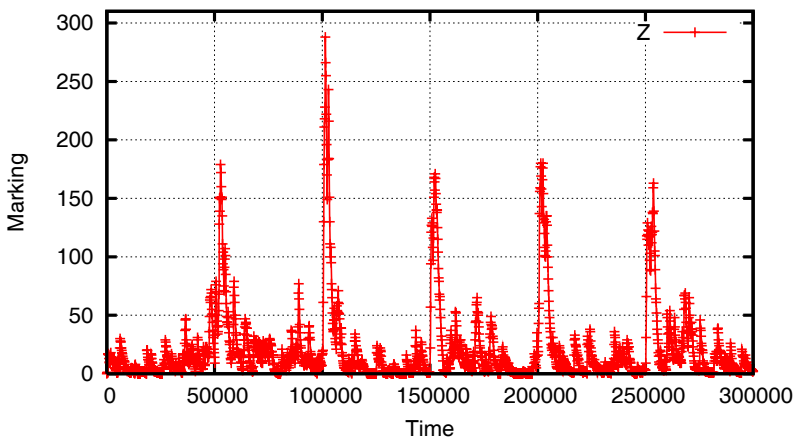

Abbildung 6.19: Simulationsergebnis vom Lac Operon-Modell: Z.

Kapitel 7

Zusammenfassung und Ausblick

Zum Abschluss der Diplomarbeit soll eine Zusammenfassung gegeben und ein Ausblick auf weitere mögliche Aufgaben formuliert werden.

7.1 Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde ein Werkzeug implementiert, welches es ermöglicht, stochastische Petrinetze zu modellieren und zu simulieren. Als erstes wurden dabei die semantischen Grundlagen, in Form der stochastischen Prozesse, beleuchtet.

Anschließend wurde eine Definition für stochastische Petrinetze entwickelt, die den verschiedenen Aspekten der Modellierung von biochemische Netzwerken Rechnung trägt. So wurden zum Beispiel spezielle Ratenfunktionen für die stochastische Massenwirkungskinetik und feste zeitbewertete Transitionen für zeitpunktgenaue Netzmanipulationen vorgestellt. Dabei wurde aufgezeigt, das die zusätzlich eingeführten Komponenten nicht dazuführen, das sich die in Snoopy erstellten Netzwerke aus den anerkannten Netzklassen $\mathcal{DSPN}$ bzw. $e\mathcal{DSPN}$ heraus bewegen.

Für die stochastische Simulation in Snoopy wurde eine angepasste Version der direkten Gillespie-Methode implementiert.

Anhand biochemischer Modelle wurde die Anwendung der neuen Petrinetzklasse in Snoopy aufgezeigt.

7.2 Ausblick

Ein wesentlicher Ansatzpunkt für weitere Betrachtungen im Umfeld der stochastischen Modellierung von biochemischen Netzwerken, ist die Untersuchung der Auswirkungen von deterministischen Transitionen, auf die Struktur der modellierten stochastischen Prozesse. Besonders interessant wären diese Betrachtungen für die Beantwortung der Frage nach der grundsätzlichen Analysefähigkeit, bezüglich struktureller und temporallogischer Eigenschaften solcher Netzwerke.

Mögliche weitergehende Aufgaben am Werkzeug Snoopy sind:

- die Implementierung von approximativen Simulationsverfahren,
- eine automatische Auswahl von Simulationsverfahren, auf Basis der vorliegenden Netzeigenschaften,
- die Darstellung mehrerer Simulationsläufe in einem Plot und
- der Export der Plots als Grafik.

Anhang A

Implementierungsstruktur

Dieses Kapitel beschreibt die Struktur des entwickelten Werkzeuges und die zur Verfügung stehenden Einstellungsmöglichkeiten, der in Abschnitt (5.2) benannten Leistungsmerkmale. Darüber hinaus sollen dem Anwender die wesentlichen Dialoge und Elemente der Benutzerschnittstelle näher gebracht werden. Dabei wird sich auf die implementierte Netzklasse der stochastischen Petrinetze beschränkt.

A.1 Netzklasse Stochastic Petri Net

Die neu erstellte Netzklasse heißt *Stochastic Petri Net*. Sie ist äquivalent mit der Klasse $\mathcal{DSPN}_{\mathcal{S}}$, welche in Unterabschnitt (3.4.2) definiert wurde.

Folgende Strukturelemente stehen zur Modellierung bereit:

- Knotenarten:
  - Place
  - Transition
  - Parameter
  - Coarse Place
  - Coarse Transition
  - Coarse Parameter
- Kantenarten:
  - Edge
  - Inhibitor Edge
  - Read Edge
- Zusätzliche Komponenten zur Modellierung:
  - LookUp-Table

  - Parameter
  - Coarse Parameter

- Zusätzliche Komponenten zur Ergebnisdarstellung:

  - Table
  - Plot

A.1.1 Stochastische Petrinetzknoten

In diesem Abschnitt werden die verschiedenen Knotenarten der neuen Netzklasse im Detail besprochen. Sie stellen, neben den Kanten, die Strukturelemente des Netzwerkes dar. Es wird vor allem auf die Attribute der Elemente und deren Manipulationsmöglichkeiten eingegangen.

Plätze

Der Knotentyp **Place** besitzt die Attribute **ID**, **Name**, **Markierungen**, **Logisch** und **Kommentar**. Markierungen können in Konfigurationssätzen organisiert werden, siehe Unterabschnitt (5.2.3).

Zur besseren Übersichtlichkeit von komplizierten Netzstrukturen wird in Snoopy die Möglichkeit angeboten, von einem Platz mehrere grafische Repräsentationen zu erstellen. Hierfür müssen alle Knoten denselben Namen besitzen und es muss jeweils die Eigenschaft **Logisch** aktiviert sein. Logische Plätze werden grau dargestellt.

Ebenfalls werden grafische Gestaltungsmöglichkeiten und ein Kommentarfeld angeboten, die bereits aus der diskreten Petrinetzklasse bekannt sind.

Transitionen

Der Knotentyp **Transition** verfügt über die Attribute **ID**, **Name**, **Funktionen**, **Logisch** und **Kommentar**.

Es werden zur schnellen und komfortablen Verwaltung von Raten- und Semantikfunktionen Konfigurationssätze angeboten, siehe Unterabschnitt (5.2.3).

Die Eigenschaft **Logisch** für Transitionen besitzt dieselbe Semantik, wie im Fall der Plätze. Eine Transition kann somit mehrere grafische Ausprägungen besitzen.

Auch hier stehen die bekannten grafischen Gestaltungsmöglichkeiten und ein Kommentarfeld zur Verfügung.

Grobplätze und -transitionen

Das Werkzeug Snoopy verfügt über die Fähigkeit Netzelemente hierarchisch auf mehreren Ebenen zu organisieren. Hierfür werden Grobplätze und -transitionen verwendet. Sie repräsentieren ein, in der Hierarchie tiefer gelegenes Teilnetz.

Für die Verknüpfung von Netzelementen eines höheren Teilnetzes mit denen eines tieferen, müssen lediglich Kanten von einem Platz bzw. einer Transition zum entsprechenden Grobplatz oder -transition gezogen werden.

Die Knotentypen **Grobplatz** und **-transition** verfügen über die Attribute **Name** und **Kommentar**, sowie über die bekannten grafischen Gestaltungsmöglichkeiten.

A.1.2 Stochastische Petrinetzkanten

Die Netzklasse der stochastischen Petrinetze besitzt drei Arten von Kanten, die im wesentlichen aus der Klasse der erweiterten diskreten Petrinetze bekannt sind [9].

Kanten

Die Kanten sind gerichtet und können durch das Attribut **Multiplicity** ganzzahlig nichtnegativ gewichtet werden. Des weiteren ist es möglich ein **Kommentar** hinzuzufügen und es steht die Option zur grafischen Einfärbung zur Verfügung.

Inhibitorkanten

Die Semantik der Inhibitorkanten wurde in Unterabschnitt (3.4.1) beschrieben. Sie sind gerichtet und können lediglich zwischen Plätzen und Transitionen gezogen werden. Als Kantenspitze wird ein unausgefüllter Kreis verwendet.

Das Kantengewicht wird durch das Attribut **Multiplicity** quantifiziert, welches ganzzahlig und nicht-negativ ist. Wie bei den normalen Kanten ist es möglich einen **Kommentar** anzugeben und die grafische Darstellung einzufärben.

Lesekanten

Die Lesekanten wurden ebenfalls bereits in Unterabschnitt (3.4.2) vorgestellt. Auch sie können nur von Plätzen zu Transitionen gezeichnet werden. In diesem Fall wird als Kantenspitze ein ausgefüllter Kreis benutzt.

Das Kantengewicht **Multiplicity** ist ganzzahlig und nicht-negativ. Die Möglichkeit für das Hinzufügen eines **Kommentars** und die Option zur grafischen Einfärbung sind obligatorisch.

A.1.3 Zusätzliche Komponenten zur Modellierung

Neben den Strukturelementen (Plätze, Transitionen und Kanten) werden zusätzliche Komponenten angeboten, welche die Modellierung der Quantitäten des Netzwerkes erleichtern sollen.

Wertetabellen

Eine Wertetabelle stellt eine reellwertige frei definierbare Funktion dar, siehe Unterabschnitt (5.2.2). Ihre Zuordnungsvorschrift wird nicht geschlossen, in Gestalt einer Formel,

angegeben, sondern sie wird einzeln für jedes $(x, f(x))$-Paar definiert. Die hier spezifizierten Funktionen können zur Formulierung von Raten- oder Semantikfunktionen genutzt werden, siehe Abschnitt (5.2).

Für den Fall, das während einer Berechnung ein nicht explizit definierter Funktionswert angefordert wird, muss ein fester **Standardrückgabewert** festgelegt werden. Dies bedeutet, das alle nicht explizit definierten Werte x aus dem Wertebereich der reellen Zahlen auf den Standardrückgabewert abgebildet werden.

Parameter

Der Knotentyp **Parameter** repräsentiert einen numerischen Wert, der für die Konstruktion von Ratenfunktionen eingesetzt werden kann. Er verfügt über die Attribute **ID**, **Name** und **Parameter**.

Mithilfe der Definition von Parameter-Konfigurationssätzen können verschiedene reellwertige Ausprägungen für einen Parameter verwaltet werden.

Grobparameter

Zur besseren Übersichtlichkeit können Parameter mittels eines Grobparameter-Knotens thematisch gruppiert und in eine eigene Netzebene verschoben werden.

A.1.4 Zusätzliche Komponenten zur Ergebnisdarstellung

Die Ergebnisse einer Simulationsanfrage werden entweder tabellarisch oder grafisch als Plot präsentiert. Damit bestimmte Platzmengen und Zeitintervalle isoliert betrachtet werden können, wurden die Knotentypen **Tabelle** und **Plot** integriert. Sie werden direkt, innerhalb des zentralen Simulationsdialoges (Abschnitt (A.2)), verwaltet und besitzen keine grafische Ausprägung als Netzelement.

Eine Haupttabelle bzw. ein Hauptplot sind standardmäßig für jedes Netzwerk definiert. Sie können nicht vom Anwender umbenannt oder gelöscht werden.

Tabellen

Die Komponente **Tabelle** besitzt die Attribute **Name** und **Kommentar**. Mithilfe der Auswahlliste **Platzauswahl** werden die Plätze markiert, die in der Tabelle als Spalten aufgeführt werden sollen (Abb. (A.1)).

Plot

Die Komponente **Plot** besitzt ebenfalls die Attribute **Name**, **Kommentar** und **Platzauswahl**. Es besteht die Möglichkeit, ein automatisches oder festes Justieren des Darstellungsbereiches festzulegen. Die Option **Justierung** gibt diese Auswahl vor. Ist die Auswahl auf fest gestellt, werden die Darstellungsinterval Plot-Koordinatensystemes entsprechend angepasst.

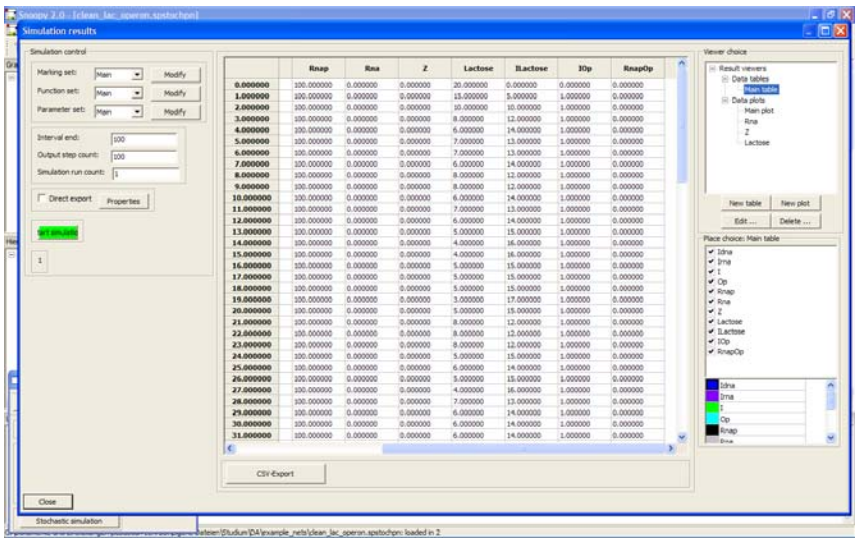

Abbildung A.1: Simulationsdialog mit Ergebnisdarstellung als Tabelle.

A.1.5 Konfigurationssätze

Mit dem Konzept der Konfigurationssätze soll dem Anwender die Möglichkeit gegeben werden, ein strukturell unverändertes Netz schnell und komfortabel mit flexiblen Markierungen, Ratenfunktionen und Parametern zu animieren und zu simulieren.

So ist es zum Beispiel möglich, mehrere Anfangsmarkierungen für ein Netzwerk anzulegen. Besteht nun der Wunsch, das Verhalten des Netzwerkes für verschiedene Anfangsmarkierungen unter veränderten Ratenfunktionen zu beobachten, ist es nicht nötig im Editiermodus die einzelnen Platzmarkierungen anzupassen.

Die Konfigurationssätze werden in den entsprechenden Übersichtsdialogen für Markierungen, Funktionen und Parametern verwaltet, siehe Abbildung (A.2). Standardmäßig ist für jedes Netzwerk ein Hauptkonfigurationssatz für Markierungen, Ratenfunktionen und Parametern angelegt, der nicht umbenannt oder gelöscht werden kann.

In der grafischen Repräsentation des Netzwerkes werden stets die Ausprägungen der entsprechenden Hauptkonfigurationssätze angezeigt.

A.1.6 Eingabeassistent für Raten- und Semantikfunktionen

Der Eingabeassistent für die Raten- und Semantikfunktionen (Abb. (A.3)) soll dem Anwender eine Unterstützung für die korrekte Formulierung von Funktionen bieten. Es werden nur Netzkomponenten und mathematischen Funktionen angezeigt, die zur Bildung der entsprechenden Formel herangezogen werden dürfen.

So werden alle Vorplätze der jeweiligen Transition, alle definierten Parameter, alle definierten Wertetabellen, sowie die vorhandenen mathematischen Funktionen in einer Übersicht angeboten.

Die vordefinierten Funktionen sind in Tabelle (A.1) aufgelistet. Die vordefinierten Parameter können in Tabelle (A.2) nach gelesen werden.

Die vorgegebenen Parameter sind vor allem für die Definition von Zeitintervallen be-

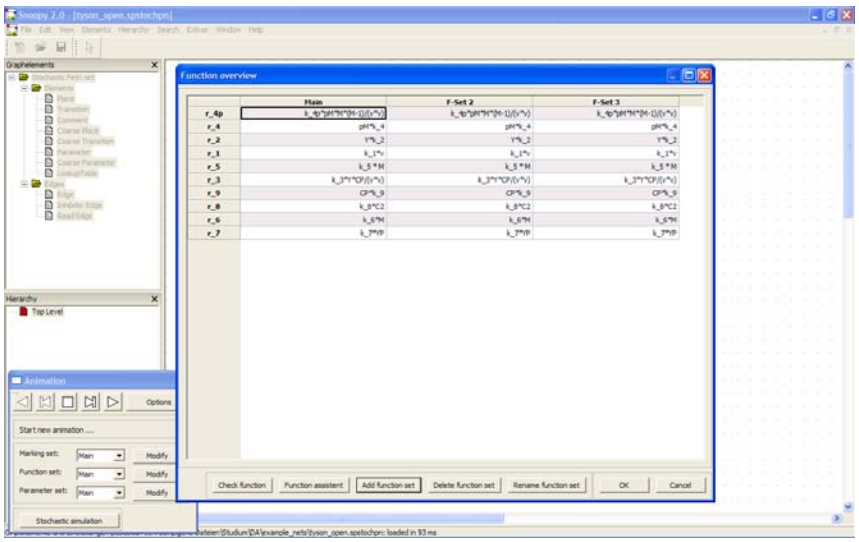

Abbildung A.2: Konfigurationssatzdialog für Raten- und Semantikfunktionen.

stimmt. Angenommen eine feste zeitbewertete Transition $t_{fixedTimed}$ ist mit der Semantikfunktion *FixedTimedFiring_Periodic(0, 5, (_SimEnd-30))* angegeben. Dann versucht $t_{fixedTimed}$ aller 5 Zeiteinheiten zu schalten, bis 30 Zeiteinheiten vor Simulationsende. Werden nun Simulationsanfragen mit verschiedenen Simulationslängen an das System gestellt, passt sich das Zeitintervall der festen zeitbewerteten Transition $t\_fixedTimed$ automatisch an.

A.2 Simulation

Die Steuerung einer Simulationsanfrage und die Darstellung des entsprechenden Ergebnisses befindet sich im Simulationsdialog (Abb. (A.3)). Dabei ist im linken Bereich die Eingabe für die Anfrageparameter, im mittleren Teil die Ergebnisdarstellung und im rechten die Verwaltung der Darstellungskomponenten angeordnet. Auf die beiden letzteren Bereiche wird erst im nächsten Abschnitt eingegangen.

Wie im Unterabschnitt (5.2.3) beschrieben, können jeder Simulationsanfrage verschiedene Konfigurationssätze, bezüglich Markierung, Ratenfunktionen und Parametern, zu Grunde gelegt werden.

Folgende weitere Einstellungen sind möglich:

- Zeitende für Simulationsintervall
- Anzahl der Ausgabezeitintervalle
- Anzahl der zu simulierenden Läufe

Je größer die Anzahl der Ausgabezeitintervalle gewählt wird, umso feiner ist die Unterteilung des gesamten Simulationsintervalles für die Ausgabe. Die Beispiele aus Tabelle

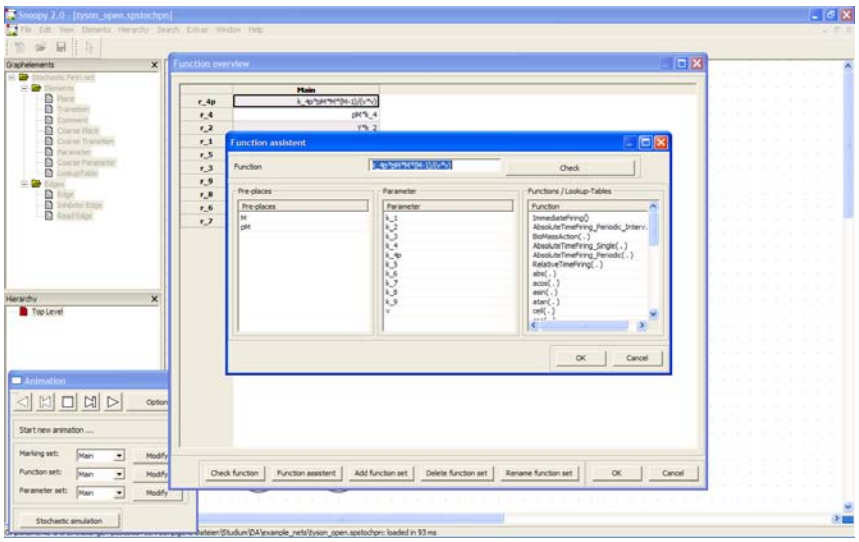

Abbildung A.3: Dialog des Eingabeassistents für Raten- und Semantikfunktionen.

(A.3) sollen die Beziehung zwischen Simulationszeitintervallgröße und Anzahl der Ausgabezeitintervalle verdeutlichen.

Eine größere Anzahl von Ausgabeintervallen verursacht eine Verschlechterung des Laufzeitverhaltens, da innerhalb eines Simulationslaufes mehr Ergebnisausgabepunkte erzeugt werden müssen.

A.3 Darstellung der Simulationsergebnisse

Im mittleren und rechten Teil des Simulationsdialoges befinden sich zum einen, der Bereich für die aktuelle Ausgabetabelle oder den aktuellen Plot und zum anderen, die Verwaltung der Ergebnisdarstellungskomponenten.

Die Tabellen und Plots sind durch einfaches Anklicken des entsprechenden Eintrages im Verwaltungsbaum auswählbar. Unter dem Verwaltungsbaum ist eine Auswahlliste für die angezeigten Plätze der aktuellen Darstellungskomponente zu finden.

Der Darstellungsbereich für einen momentan geladenen Plot lässt sich, mittels den darunter liegenden Schaltknöpfen, verändern.

Folgende Funktionen sind mit den Schaltknöpfen verbunden (von links nach rechts):

- Wertachse stauchen
- Wertachse strecken
- Zeitachse stauchen
- Zeitachse strecken
- Hinein zoomen

Funktion	**Bedeutung**
BioMassAction(.)	Ratenfunktion für Molekülinterpretation von Marken, siehe Definition (3.9)
BioLevelInterpretation(.)	Ratenfunktion für Intervallinterpretation von Marken, siehe Definition (3.10)
ImmediateFiring()	unmittelbare Transition, siehe Unterabschnitt (3.4.1)
TimedFiring(.)	zeitbewertete Transition, siehe Unterabschnitt (3.4.2)
FixedTimedFiring_Single(.)	feste zeitbewertete Transition mit einmaligem Schaltversuch, siehe Unterabschnitt (5.2.5)
FixedTimedFiring_Periodic(. , . , .)	feste zeitbewertete Transition mit periodischen Schaltversuchen, siehe Unterabschnitt (5.2.5)
abs(.)	Absolutwert
acos(.)	Arkuskosinus-Funktion
asin(.)	Arkussinus-Funktion
atan(.)	Arkustangens-Funktion
ceil(.)	Aufrunden
cos(.)	Kosinus-Funktion
exp(.)	Exponential-Funktion
sin(.)	Sinus-Funktion
sqr(.)	Quadrat
sqrt(.)	Wurzel
tan(.)	Tangens-Funktion
floor(.)	Abrunden
log(.)	Logarithmus zur Basis e
log10(.)	Logarithmus zur Basis 10
pow(.)	Exponent

Tabelle A.1: Angebotene Raten-/Semantikfunktionen und mathematischen Funktionen.

- Heraus zoomen
- Zentrieren des Darstellungsbereiches

Die Kurven werden zwischen den exakt berechneten Punkten mit Geraden interpoliert. Die ermittelten Punkte werden auf einer Kurve mit einem kleinen Kreis gekennzeichnet. Mittels Anklicken einer dieser Kreise, wird der jeweiligen Wert und der Platzname der Kurve neben den Schaltknöpfen angezeigt.

A.4 Export

Bezüglich des Exports, wird zwischen dem Abspeichern von Simulationsergebnissen und dem Konvertieren von Netzwerken in andere Netzklassen unterschieden.

Parameter	Bedeutung
_SimStart	Anfangszeitpunkt des Simulationsintervalles, besitzt den festen Wert Null
_SimEnd	Endzeitpunkt des Simulationsintervalles
_SimTime	aktueller Zeitpunkt des Simulationslaufes

Tabelle A.2: Vordefinierte Zeitparameter.

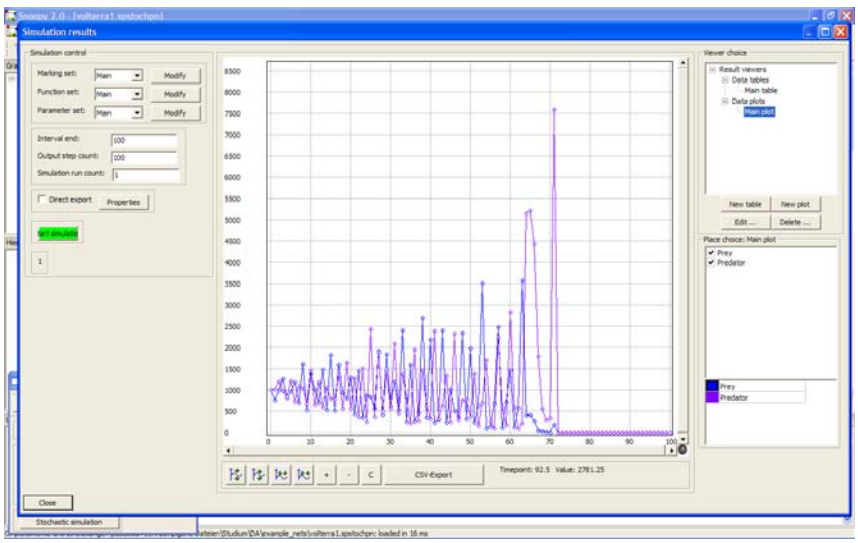

Abbildung A.4: Simulationsdialog mit Ergebnisdarstellung als Plot.

CSV-Export

Auch der Export von Simulationsergebnissen wird, innerhalb des Simulationsdialoges, gesteuert. Im linken Bereich befindet sich eine Auswahloption für die Aktivierung des direkten CSV-Export. Ist die Option markiert, wird automatisch für jedes Ergebnis einer jeden Simulationsanfrage eine Exportdatei erzeugt. Im dazugehörigen Eigenschaftsdialog kann der Name der Exportdatei und das zu benutzende Trennzeichen für die Erzeugung der Tabellenstruktur spezifiziert werden. Für den direkten CSV-Export werden stets die Platzauswahl und das Zeitintervall der Haupttabelle benutzt.

Neben dem automatischen CSV-Export können auch die Ergebnisdaten jeder Darstellungskomponente, Tabelle oder Plot, einzeln exportiert werden. Hierbei wird explizit ein Dateiname, sowie das Trennzeichen festgelegt. Platzauswahl und Zeitintervall werden aus der Parameterisierung der Darstellungskomponente übernommen.

Gesamtintervall	Anzahl der Intervalle	Größe der Intervalle
50	50	Ausgabe pro 1 Zeiteinheit
50	100	Ausgabe pro 0.5 Zeiteinheiten
50	25	Ausgabe pro 2 Zeiteinheiten

Tabelle A.3: Beziehungen zwischen Anzahl und Größe der Ausgabeintervalle

Export in andere Netzklassen

Ein stochastisches Petrinetz kann in ein diskretes oder kontinuierliches Petrinetz umgewandelt werden. Dies geschieht durch die allgemeine Exportfunktion von Snoopy. Das Konzept der Konfigurationssätze wird bisher für die Netzklassen der diskreten und kontinuierlichen Petrinetze nur eingeschränkt unterstützt. Aus diesem Grund kann jeweils nur ein Konfigurationssatz für Markierungen, Funktionen und Parametern ausgewählt werden.

Die Informationen über Funktionen und Parameter sind für diskrete Netzwerke nicht relevant und werden ignoriert.

Stochastische Raten- und Semantikfunktionen werden im Kontext der kontinuierlichen Petrinetze als Schaltgeschwindigkeitsfunktionen (siehe [28]) interpretiert. Es wird jedoch im Allgemeinen zu Fehlern bei der Interpretation kommen, da eine Vielzahl der speziellen stochastischen Raten- und Semantikfunktionen in kontinuierlichen Petrinetzen unbekannt sind. Parameter werden automatisch übernommen.

Import in andere Netzklassen

Die Transformation eines diskreten Petrinetzes in ein stochastisches Petrinetz kann, mittels der Exportfunktion des diskreten Petrinetzes, durchgeführt werden. Die Markierungen des diskreten Netzwerkes werden als Hauptkonfigurationssatz übernommen. Die Ratenfunktionen in den stochastischen Petrinetzen werden als konstante Funktionen mit Wert Eins definiert.

Export in andere Petrinetzwerkzeuge oder -formate

Die Tabelle (A.4) enthält alle Programme oder Formate, in die ein Export der Netzwerke möglich ist.

Programme
EPS
MIF
Xfig
APNN
INA
Lola
PEP (low level, without layout)
Prod
TINA

Tabelle A.4: Liste der Programme oder Formate, in die exportiert werden kann.

Anhang B

Petrinetze und Simulationsergebnisse

B.1 Jäger-Beute-Modell von Lotka-Volterra

Reaktionen

$$\begin{aligned} \textbf{reproduction\_of\_prey} : \text{Prey} &\longrightarrow 2 * \text{Prey} \\ \textbf{predator\_death} : \text{Predator} &\longrightarrow \emptyset \\ \textbf{consumption\_of\_prey} : \text{Prey} + \text{Predator} &\longrightarrow 2 * \text{Predator} \end{aligned}$$

Anfangsmarkierung

$$\begin{aligned} \text{Prey} &= 1000 \\ \text{Predator} &= 1000 \end{aligned}$$

Ratenfunktionen

$$\begin{aligned} \textbf{reproduction\_of\_prey} &= \text{BioMassAction}(c_1) \\ \textbf{predator\_death} &= \text{BioMassAction}(c_2) \\ \textbf{consumption\_of\_prey} &= \text{BioMassAction}(c_3) \end{aligned}$$

Parameter

$$
\begin{aligned}
c_1 &= 10 \\
c_2 &= 10 \\
c_3 &= 0.01
\end{aligned}
$$

Simulationsergebnisse

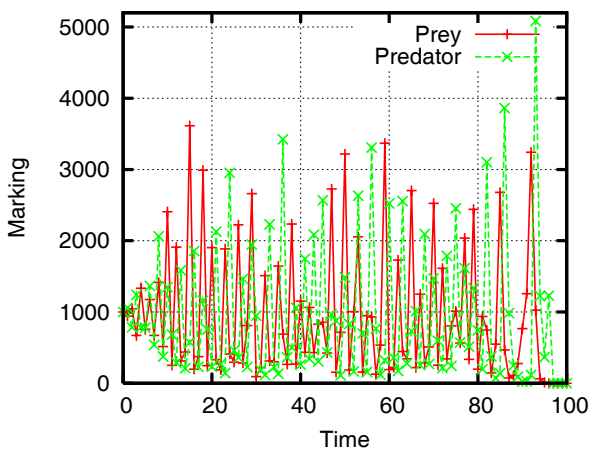

Abbildung B.1: Simulationsergebnisse des Jäger-Beute-Modelles für einen Lauf.

B.2 Zellzyklus von Tyson

Reaktionen

$$\begin{aligned}
\mathbf{r_1} &: \emptyset \longrightarrow \mathrm{Y} \\
\mathbf{r_2} &: \mathrm{Y} \longrightarrow \emptyset \\
\mathbf{r_3} &: \mathrm{CP} + \mathrm{Y} \longrightarrow \mathrm{pM} \\
\mathbf{r_4} &: \mathrm{pM} \longrightarrow \mathrm{M} \\
\mathbf{r_{4p}} &: \mathrm{pM} + 2 * \mathrm{M} \longrightarrow 3 * \mathrm{M}
\end{aligned}$$

$$\begin{aligned}
\mathbf{r_5} &: \mathrm{M} \longrightarrow \mathrm{C2} + \mathrm{YP} \\
\mathbf{r_6} &: \mathrm{YP} \longrightarrow \emptyset \\
\mathbf{r_7} &: \emptyset \longrightarrow \mathrm{YP} \\
\mathbf{r_8} &: \mathrm{C2} \longrightarrow \mathrm{CP} \\
\mathbf{r_9} &: \mathrm{CP} \longrightarrow \mathrm{C2}
\end{aligned}$$

Anfangsmarkierung

$$\begin{aligned}
\mathrm{Y} &= 0 \\
\mathrm{YP} &= 0
\end{aligned}$$

$$\begin{aligned}
\mathrm{pM} &= 0 \\
\mathrm{M} &= 0
\end{aligned}$$

$$\begin{aligned}
\mathrm{CP} &= 0 \\
\mathrm{C2} &= 1000
\end{aligned}$$

Ratenfunktionen

$$\begin{aligned}
\mathbf{r_1} &= \mathrm{c}_1 * \mathrm{v} \\
\mathbf{r_2} &= \mathrm{c}_2 * \mathrm{Y} \\
\mathbf{r_3} &= \mathrm{c}_3 * \mathrm{Y} * \mathrm{CP} * \mathrm{v}^{-2} \\
\mathbf{r_4} &= \mathrm{c}_4 * \mathrm{pM} \\
\mathbf{r_4} &= \mathrm{c}_{4\mathrm{p}} * \mathrm{pM} * \mathrm{M} * (\mathrm{M} - 1) * \mathrm{v}^{-2}
\end{aligned}$$

$$\begin{aligned}
\mathbf{r_5} &= \mathrm{c}_5 * \mathrm{M} \\
\mathbf{r_6} &= \mathrm{c}_6 * \mathrm{M} \\
\mathbf{r_7} &= \mathrm{c}_7 * \mathrm{YP} \\
\mathbf{r_8} &= \mathrm{c}_8 * \mathrm{C2} \\
\mathbf{r_9} &= \mathrm{c}_9 * \mathrm{CP}
\end{aligned}$$

Parameter

$$\begin{aligned}
\mathrm{c}_1 &= 0.015 \\
\mathrm{c}_2 &= 0 \\
\mathrm{c}_3 &= 200 \\
\mathrm{c}_4 &= 0.018
\end{aligned}$$

$$\begin{aligned}
\mathrm{c}_{4\mathrm{p}} &= 180 \\
\mathrm{c}_5 &= 0 \\
\mathrm{c}_6 &= 1 \\
\mathrm{c}_7 &= 0.6
\end{aligned}$$

$$\begin{aligned}
\mathrm{c}_8 &= 100 \\
\mathrm{c}_9 &= 100 \\
\mathrm{v} &= 1000
\end{aligned}$$

Simulationsergebnisse

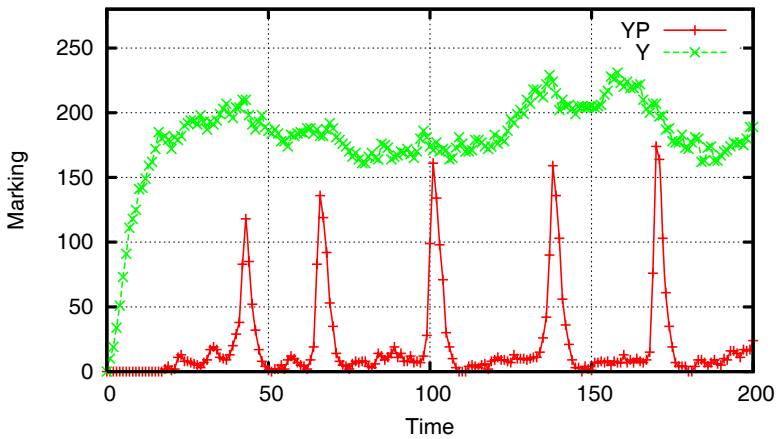

Abbildung B.2: Simulationsergebnis des Zellzyklus-Modelles von Tyson: YP, Y.

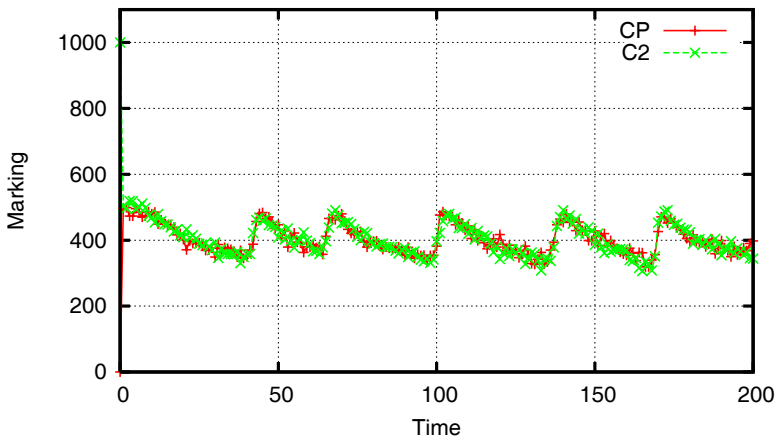

Abbildung B.3: Simulationsergebnis des Zellzyklus-Modelles von Tyson: CP, C2.

Weitere Simulationsergebnisse für die Stoffe pM und M sind in der Abbildung (6.15) zu sehen.

B.3 MAPK-Kaskade von Levchenko

Reaktionen

$$
\begin{aligned}
\mathbf{r_1} &: \text{Raf} + \text{RasGTP} \longrightarrow \text{Raf\_RasGTP} \\
\mathbf{r_2} &: \text{Raf\_RasGTP} \longrightarrow \text{Raf} + \text{RasGTP} \\
\mathbf{r_3} &: \text{Raf\_RasGTP} \longrightarrow \text{RasGTP} + \text{RafP} \\
\mathbf{r_4} &: \text{RafP} + \text{Phase1} \longrightarrow \text{RafP\_Phase1} \\
\mathbf{r_5} &: \text{RafP\_Phase1} \longrightarrow \text{RafP} + \text{Phase1} \\
\mathbf{r_6} &: \text{RafP\_Phase1} \longrightarrow \text{Raf} + \text{Phase1} \\
\mathbf{r_7} &: \text{MEK} + \text{RafP} \longrightarrow \text{MEK\_RafP} \\
\mathbf{r_8} &: \text{MEK\_RafP} \longrightarrow \text{MEK} + \text{RafP} \\
\mathbf{r_9} &: \text{MEK\_RafP} \longrightarrow \text{MEKP} + \text{RafP} \\
\mathbf{r_{10}} &: \text{MEKP} + \text{RafP} \longrightarrow \text{MEKP\_RafP} \\
\mathbf{r_{11}} &: \text{MEKP\_RafP} \longrightarrow \text{MEKP} + \text{RafP} \\
\mathbf{r_{12}} &: \text{MEKP\_RafP} \longrightarrow \text{MEKPP} + \text{RafP} \\
\mathbf{r_{13}} &: \text{MEKPP} + \text{Phase2} \longrightarrow \text{MEKPP\_Phase2} \\
\mathbf{r_{14}} &: \text{MEKPP\_Phase2} \longrightarrow \text{MEKPP} + \text{Phase2} \\
\mathbf{r_{15}} &: \text{MEKPP\_Phase2} \longrightarrow \text{MEKP} + \text{Phase2} \\
\mathbf{r_{16}} &: \text{MEKP} + \text{Phase2} \longrightarrow \text{MEKP\_Phase2} \\
\mathbf{r_{17}} &: \text{MEKP\_Phase2} \longrightarrow \text{MEKP} + \text{Phase2} \\
\mathbf{r_{18}} &: \text{MEKP\_Phase2} \longrightarrow \text{MEK} + \text{Phase2} \\
\mathbf{r_{19}} &: \text{ERK} + \text{MEKPP} \longrightarrow \text{ERK\_MEKPP} \\
\mathbf{r_{20}} &: \text{ERK\_MEKPP} \longrightarrow \text{ERK} + \text{MEKPP} \\
\mathbf{r_{21}} &: \text{ERK\_MEKPP} \longrightarrow \text{ERKP} + \text{MEKPP} \\
\mathbf{r_{22}} &: \text{ERKP} + \text{MEKPP} \longrightarrow \text{ERKP\_MEKPP} \\
\mathbf{r_{23}} &: \text{ERKP\_MEKPP} \longrightarrow \text{ERKP} + \text{MEKPP} \\
\mathbf{r_{24}} &: \text{ERKP\_MEKPP} \longrightarrow \text{ERKPP} + \text{MEKPP} \\
\mathbf{r_{25}} &: \text{ERKPP} + \text{Phase3} \longrightarrow \text{ERKPP\_Phase3} \\
\mathbf{r_{26}} &: \text{ERKPP\_Phase3} \longrightarrow \text{ERKPP} + \text{Phase3} \\
\mathbf{r_{27}} &: \text{ERKPP\_Phase3} \longrightarrow \text{ERKP} + \text{Phase3} \\
\mathbf{r_{28}} &: \text{ERKP} + \text{Phase3} \longrightarrow \text{ERKP\_Phase3} \\
\mathbf{r_{29}} &: \text{ERKP\_Phase3} \longrightarrow \text{ERKP} + \text{Phase3} \\
\mathbf{r_{30}} &: \text{ERKP\_Phase3} \longrightarrow \text{ERK} + \text{Phase3}
\end{aligned}
$$

Anfangsmarkierung

RasGTP	= 1	MEKP_RafP	= 0	ERKPP	= 0
Raf	= 4	MEKPP	= 0	ERKPP_Phase3	= 0
Raf_RasGTP	= 0	MEKPP_Phase2	= 0	ERKP_Phase3	= 0
RafP	= 0	MEKP_Phase2	= 0	Phase1	= 3
RafP_Phase1	= 0	ERK	= 3	Phase2	= 2
MEK	= 2	ERK_MEKPP	= 0	Phase3	= 3
MEK_RafP	= 0	ERKP	= 0	N	= 4
MEKP	= 0	ERKP_MEKPP	= 0		

Ratenfunktionen

Für jede Reaktion gilt $\mathbf{r_i} = \text{BioLevelInterpretation}(\text{k}_\text{i}, \text{N})$, wobei $i \in \{1, \ldots, 30\}$.

Parameter

k_1	= 1.0	k_{11}	= 0.4	k_{21}	= 0.1
k_2	= 0.4	k_{12}	= 0.1	k_{22}	= 20.0
k_3	= 0.1	k_{13}	= 10.0	k_{23}	= 0.6
k_4	= 0.5	k_{14}	= 0.8	k_{24}	= 0.1
k_5	= 0.5	k_{15}	= 0.1	k_{25}	= 5.0
k_6	= 0.1	k_{16}	= 10.0	k_{26}	= 0.4
k_7	= 3.3	k_{17}	= 0.8	k_{27}	= 0.1
k_8	= 0.42	k_{18}	= 0.1	k_{28}	= 5.0
k_9	= 0.1	k_{19}	= 20.0	k_{29}	= 0.4
k_{10}	= 3.3	k_{20}	= 0.6	k_{30}	= 0.1

Simulationsergebnisse

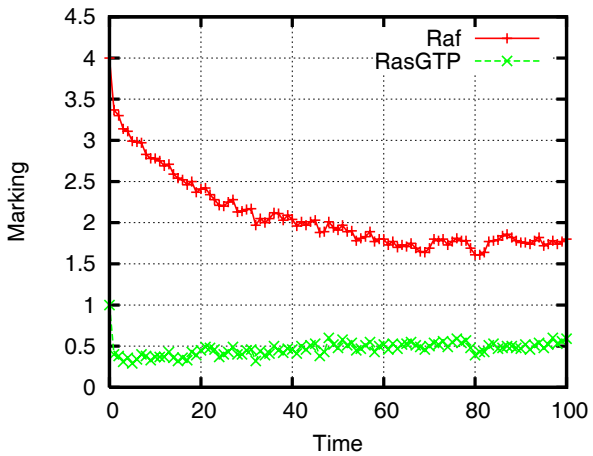

Abbildung B.4: Simulationsergebnis des MAPK-Modelles von Levchenko: Raf, RasGTP.

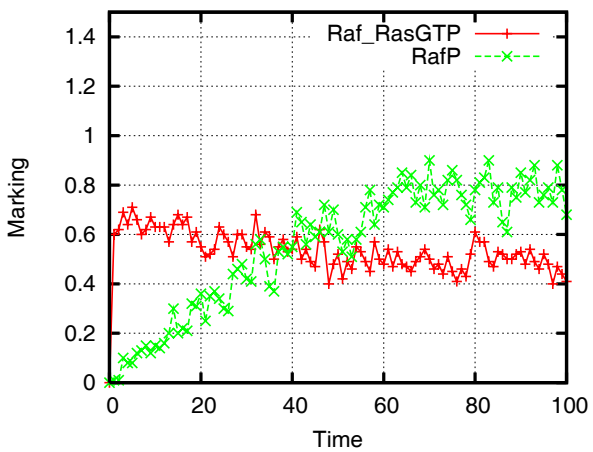

Abbildung B.5: Simulationsergebnis des MAPK-Modelles von Levchenko: Raf_RasGTP, RafP.

B.4 Lac Operon

Reaktionen

$$
\begin{aligned}
\textbf{InhibitorTranscription} &: \text{Idna} \longrightarrow \text{Idna} + \text{Irna} \\
\textbf{InhibitorTranslation} &: \text{Irna} \longrightarrow \text{Irna} + \text{I} \\
\textbf{LactoseInhibitorBinding} &: \text{I} + \text{Lactose} \longrightarrow \text{ILactose} \\
\textbf{LactoseInhibitorDissociation} &: \text{ILactose} \longrightarrow \text{I} + \text{Lactose} \\
\textbf{InhibitorBinding} &: \text{I} + \text{Op} \longrightarrow \text{IOp} \\
\textbf{InhibitorDissociation} &: \text{IOp} \longrightarrow \text{I} + \text{Op} \\
\textbf{RnapBinding} &: \text{Op} + \text{Rna} \longrightarrow \text{RnaOp} \\
\textbf{RnapDissociation} &: \text{RnapOp} \longrightarrow \text{Op} + \text{Rnap} \\
\textbf{Transcription} &: \text{RnapOp} \longrightarrow \text{Op} + \text{Rnap} + \text{Rna} \\
\textbf{Translation} &: \text{Rna} \longrightarrow \text{Rna} + \text{Z} \\
\textbf{Conversion} &: \text{Lactose} + \text{Z} \longrightarrow \text{Z} \\
\textbf{InhibitorRnaDegradation} &: \text{Irna} \longrightarrow \emptyset \\
\textbf{InhibitorDegradation} &: \text{I} \longrightarrow \emptyset \\
\textbf{LactoseInhibitorDegradation} &: \text{ILactose} \longrightarrow \text{Lactose} \\
\textbf{RnaDegradation} &: \text{Rna} \longrightarrow \emptyset \\
\textbf{ZDegradation} &: \text{Z} \longrightarrow \emptyset \\
\textbf{Intervention} &: \emptyset \longrightarrow 10000 * \text{Lactose}
\end{aligned}
$$

Anfangsmarkierung

$$
\begin{aligned}
\text{Idna} &= 1 \\
\text{Irna} &= 0 \\
\text{I} &= 50 \\
\text{Op} &= 1
\end{aligned}
\qquad
\begin{aligned}
\text{Rnap} &= 100 \\
\text{Rna} &= 0 \\
\text{Z} &= 0 \\
\text{Lactose} &= 20
\end{aligned}
\qquad
\begin{aligned}
\text{ILactose} &= 0 \\
\text{IOp} &= 0 \\
\text{RnapOp} &= 0
\end{aligned}
$$

Ratenfunktionen

$$
\begin{aligned}
\textbf{InhibitorTranscription} &= \text{BioMassAction}(c_1)\\
\textbf{InhibitorTranslation} &= \text{BioMassAction}(c_2)\\
\textbf{LactoseInhibitorBinding} &= \text{BioMassAction}(c_3)\\
\textbf{LactoseInhibitorDissociation} &= \text{BioMassAction}(c_4)\\
\textbf{InhibitorBinding} &= \text{BioMassAction}(c_5)\\
\textbf{InhibitorDissociation} &= \text{BioMassAction}(c_6)\\
\textbf{RnapBinding} &= \text{BioMassAction}(c_7)\\
\textbf{RnapDissociation} &= \text{BioMassAction}(c_8)\\
\textbf{Transcription} &= \text{BioMassAction}(c_9)\\
\textbf{Translation} &= \text{BioMassAction}(c_{10})\\
\textbf{Conversion} &= \text{BioMassAction}(c_{11})\\
\textbf{InhibitorRnaDegradation} &= \text{BioMassAction}(c_{12})\\
\textbf{InhibitorDegradation} &= \text{BioMassAction}(c_{13})\\
\textbf{LactoseInhibitorDegradation} &= \text{BioMassAction}(c_{13})\\
\textbf{RnaDegradation} &= \text{BioMassAction}(c_{14})\\
\textbf{ZDegradation} &= \text{BioMassAction}(c_{15})\\
\textbf{Intervention} &= \text{FixedTimedFiring\_Periodic}(0, 50000, \text{\_SimEnd})
\end{aligned}
$$

Parameter

$$
\begin{aligned}
c_1 &= 0.02 & c_6 &= 0.01 & c_{11} &= 0.0005\\
c_2 &= 0.1 & c_7 &= 0.1 & c_{12} &= 0.01\\
c_3 &= 0.005 & c_8 &= 0.01 & c_{13} &= 0.002\\
c_4 &= 0.1 & c_9 &= 0.03 & c_{14} &= 0.01\\
c_5 &= 1 & c_{10} &= 0.1 & c_{15} &= 0.001
\end{aligned}
$$

Simulationsergebnisse

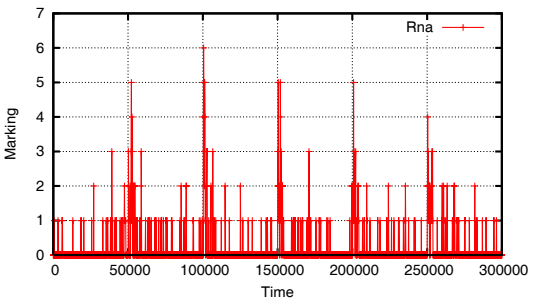

Abbildung B.6: Simulationsergebnis vom Lac Operon-Modell: Rna (einzelner Lauf).

Weitere Simulationsergebnisse für die Stoffe *Lactose* und *Z* sind in den Abbildungen (6.18) und (6.19) zu sehen.

B.5 Berechnungszeiten

In diesem Abschnitt sollen die Berechnungszeiten für die eben beschriebenen Modelle aufgeführt werden, siehe Tabelle (B.1) Die Simulation wurde unter Microsoft Windows XP auf einem AMD Sempron Prozessor 3000+ mit 896 MB Hauptspeicher durchgeführt. Die Berechnungszeiten werden in Sekunden angegeben.

Modell	Intervallende	Einzelner Lauf	100 Läufe
Jäger-Beute-Modell von Lotka-Volterra	100	6	598
Zellzyklus von Tyson	100	35	3485
MAPK-Kaskade von Levchenko	100	1	2
Lac Operon	300000	10	980

Tabelle B.1: Berechnungszeiten für die vorgestellten Modelle in Sekunden.

Literaturverzeichnis

[1] Baumgarten, B., *Petri-Netze*. Spektrum Akademischer Verlag, 1996

[2] Bause, F.; Kritzinger, P.S., *Stochastic Petri Nets*. 2nd Edition, Vieweg, 2002

[3] Calder, M.; Vyshemirsky, V.; Orton, R.; Gilbert, D., *Analysis of Signalling Pathways using the PRISM model checker*. Computational Methods in Systems Biology 2005, LFCS, University of Edinburgh, 2005, pp. 179-190

[4] David, R.; Alla, H., *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2005

[5] Dümbgen, L., *Stochastik für Informatiker*. Springer-Verlag, 2003

[6] Evans, T.W.; Gillespie, C.S.; Wilkinson, D.J., *SBML DSMTS User Guide*. http://www.calibayes.ncl.ac.uk/Resources/dsmts/, 2006

[7] Falbe, J.; Regitz, M., *Römpp-Chemie-Lexikon*. Thieme-Verlag, 1990

[8] Fieber, M., *Entwurf und Implementierung eines generischen, adaptiven Werkzeugs zur Arbeit mit Graphen*. BTU Cottbus, Diplomarbeit, 2004

[9] German, R., *Performance analysis of communication systems with non-Markovian stochastic Petri nets*. John Wiley Sons Ltd., 2001

[10] Gibson, M.A.; Bruck, J., *Efficient exact stochastic simulation of chemical systems with many species and many channels*. J. Phys. Chem. A 104, 2000, pp. 1876-1889

[11] Gilbert, D.; Heiner, M., *From Petri Nets to Differential Equations - an Integrative Approach for Biochemical Network Analysis*. Proc. ICATPN 2006, Turku, June, LNCS 4024, pp. 181-200

[12] Gilbert D.; Heiner, M.; Lehrack, S., *A Unifying Framework for Modelling and Analysing Biochemical Pathways Using Petri Nets*. Proc. 5th International Conference on Computational Methods in Systems Biology (CMSB 2007), Edinburgh, September, LNCS/LNBI 4695, pp. 200-216.

[13] Gillespie, D.T., *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions*. J. Comput. Phys. 22, 1976, pp. 403-434

[14] Gillespie, D.T., *Exact stochastic simulation of coupled chemical reactions*. The Journal of Physical Chemistry, Volume 81(25), 1977, pp. 2340-2361

[15] Goss, P.J.E.; Peccoud, J., *Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets*. PNAS 95(12), 1998, pp. 6750-6755

[16] Haas, P.J., *Stochastic Petri nets: Modelling, Stability, Simulation*. Springer-Verlag, 2003

[17] Hucka, M.; Finney, A.; Sauro, H.M.; Bolouri, H.; Doyle, J.C.; Kitano, H., *The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models*. Bioinformatics, Vol. 19 no. 4 2003, pages 524531, 2003

[18] Knottenbelt, W.; *PIPE: Documentation.*. http://pipe2.sourceforge.net/index.html, 2007

[19] Lehrack, S., *Three Petri net approaches for Biochemical Network Analysis*. Techn. Report I-01/2006, BTU Cottbus, July 2006, 33 p.

[20] Levchenko, A.; Bruck, J.; Sternberg, P.W., *Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties*. Proc Natl Acad Sci USA, 97(11):58185823, 2000

[21] Marsan, M.A.; Balbo, G.; Conte, G.; Donatelli, S; Franceschinis, G., *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing, 1995

[22] Papula, L., *Mathematik für Ingenieure und Naturwissenschaftler Band 3*. 4. Auflage, Vieweg Sohn Verlagsgesellschaft, 2001

[23] Petri, C.A., *Kommunikation mit Automaten*. Dissertation, Universität Bonn, Institut für Instrumentelle Mathematik, 1962

[24] Priese, L.; Wimmel, H., *Theoretische Informatik: Petri-Netze*. Springer-Verlag, 2003

[25] Ramsey, S., *Dizzy User Manual.*. CompBio Group, Institute for Systems Biology, 2006

[26] Reisig, W., *Petrinetze, Eine Einführung*. Springer-Verlag, 1991

[27] Ross, S.M., *Stochastic Processes*. Wiley Series in Probability and Mathematic Statistics, 1996

[28] Scheibler, D., *Ein Werkzeug zum Entwerfen und Simulieren kontinuierlicher Petrinetze*. BTU Cottbus, Diplomarbeit, 2006

[29] Starke, P.H., *Analyse von Petri-Netz-Modellen*. Teubner-Verlag, 1990

[30] Tyson, J.J., *Modeling the Cell Division Cycle: cdc2 and Cyclin Interactions*. Proc. Nati. Acad. Sci. USA Vol. 88, pp. 7328-7332, 1991

[31] Ullah, M.; Schmidt, H.; Cho, K.-H.; Wolkenhauer, O., *Deterministic modelling and stochastic simulation of biochemical pathways using MATLAB*. IEE Proc.-Syst. Biol., Vol. 153, No. 2, March 2006, pp. 53-60

[32] Wilkinson, D.J., *Stochastic Modelling for System Biology*. CRC Press, New York, 1st Edition, 2006

[33] Zimmermann, A.; Knole, M.; *TimeNET 4.0: A Software Tool for the Performability Evaluation with Stochastic and Colored Petri Nets: User Manual*. Technische Universität Berlin, Real-Time Systems and Robotics Group, Faculty of EE&CS Technical Report 2007-13, 2007