# Zu Algorithmen der Analyse biochemischer Systeme

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Mathematik und Informatik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von
Jens Einloft
aus Hanau

Frankfurt (2016)
(D 30)

vom Fachbereich Mathematik und Informatik der

Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr. Uwe Brinkschulte

Gutachter: Prof. Dr. Ina Koch und Prof. Dr. Enrico Schleiff

Datum der Disputation: 27.09.2016

*Soli deo gloria.*

# Inhaltsverzeichnis

# Abbildungsverzeichnis

**I Modellierung biochemischer Systeme**

# Tabellenverzeichnis

# Teil I

# Modellierung biochemischer Systeme

# Kapitel 1

# Einleitung

## 1.1 Systembiologie

Lebende Organismen sind komplexe Systeme, bestehend aus grundlegenden Bausteinen des Lebens (Machado et al., 2011). Das Erforschen und Untersuchen dieser komplexen Systeme in ihrer Gesamtheit, auf zellulärer Ebene, aber auch darüber hinaus, ist Gegenstand der Systembiologie. Das Verständnis eines biologischen Systems kann sowohl durch theoretische aber auch experimentelle Ansätze verbessert werden. Ein erster Ansatz ist die Charakterisierung der Struktur des Netzwerkes aller biochemischen Prozesse und der Interaktion der Gene und welchen Einfluss diese Interaktionen auf die Funktion und Strukturen des Organismus haben. Ein zweiter Ansatz ist die Beobachtung des Systems über die Zeit und unter verschiedenen Bedingungen, um die Dynamik des Systems zu verstehen. Ein dritter Ansatz ist die Untersuchung und Modulation derjenigen Mechanismen, welche den Zustand des Systems kontrollieren. Als vierter Ansatz können die Ergebnisse der ersten drei Ansätze in mathematische Modelle übersetzt werden, mit denen anschließend die gewonnenen Informationen getestet werden oder Hypothesen aufgestellt werden können. (Kitano, 2002)

Zur Durchführung der ersten drei Punkte wurden in den letzten Jahren experimentelle Hochdurchsatz-Methoden entwickelt. Diese erlauben die Erzeugung sogenannter *-om*-Daten, wie beispielsweise das Metabolom, die Erfassung aller Metaboliten zu einem bestimmten Zeitpunk in einer Zelle oder einem Organismus (Weckwerth, 2007). Daneben existieren Methoden zum Erfassen des Transkriptoms (Wang et al., 2009), Proteoms (Nesvizhskii, 2010), Interaktoms (De Las Rivas und Fontanillo, 2010) oder der Erfassung aller aktiven Gene (Klug et al., 2006). Die Reichhaltigkeit an solchen Methoden erlaubt die Rekonstruktion vieler biologischer Modelle (Feist et al., 2009), deren Komplexität und Größe durch die zunehmende verfügbare Datenmenge immer mehr zunimmt. Solche biologischen Modelle erfassen meist einen bestimmten Aspekt des untersuchten Systems. Der Fokus kann auf der Modellierung der metabolischen Prozesse liegen (Nöthen, 2014), einen bestimmten Signaltransduktionsweg betreffen (Janes und Lauffenburger, 2013), die Regulierung der Genexpression abbilden (Galagan et al., 2013) oder die Interaktion von Proteinen

miteinander erfassen (Li et al., 2004b). Die Konstruktion mathematischer Modelle kann quantitativ oder qualitativ erfolgen. Im Folgendem wird eine Übersicht über diese beiden Arten der Modellierung gegeben.

### 1.1.1 Qualitative Modellierung

Für die qualitative Modellierung werden keine experimentell bestimmten Parameter, wie Reaktionskonstanten, benötigt. Sie kann also auch dann zum Einsatz kommen, wenn diese Informationen nicht oder nur teilweise zur Verfügung stehen. Zum qualitativen Modellieren biologischer Systeme wurden verschiedenste Methoden entwickelt.

Bei der Booleschen Modellierung besitzen die Elemente des Modells einen binären Zustand. Verbunden werden die Elemente durch aussagenlogische Funktionen. Die Auswertung dieser Funktionen entscheidet über den Zustand eines Elements (Wang et al., 2012). Entwickelt wurde diese Methode ursprünglich zur Modellierung von genregulatorischen Netzwerken (Thomas, 1973; Kauffman, 1969), und sie wird noch immer für diesen Zweck genutzt (Albert und Othmer, 2003; Thakar et al., 2007). Inzwischen existieren auch Anwendungen für metabolische Modelle (Akutsu et al., 2000) und Signaltransduktionswege (Gupta et al., 2007; Saez-Rodriguez et al., 2007). Mit Booleschen Modellen können Fließgleichgewichte gefunden werden, oder aber die Robustheit des Modells getestet werden (Li et al., 2004a). Eine Übersicht über die Methoden der Booleschen Modellierung und deren Anwendungen in der Systembiologie wird durch Wang et al. (2012) gegeben.

Ein biologisches System kann auch mit einem Bayesschen Netz (Pearl, 1988) qualitativ modelliert werden. Ein solches ist ein gerichteter azyklischer Graph, dessen Knoten Zufallsvariablen darstellen und dessen Kanten bedingte Abhängigkeiten zwischen den Variablen beschreiben. Jeder Knoten besitzt eine Wahrscheinlichkeitsverteilung, welche von den Werten seiner Eingangsknoten abhängt. Solche Modelle wurden für die Untersuchung von genregulatorischen Netzwerken (Friedman, 2004; Auliac et al., 2008) oder von Signaltransduktionswegen (Sachs et al., 2002) verwendet. Der große Nachteil dieser Methode ist, dass mit ihr keine Rückkopplungsschleifen modelliert werden können. Diese Einschränkung ist jedoch durch den Einsatz von dynamischen Bayesschen Netzen aufgehoben (Husmeier, 2003).

Ein weiterer Ansatz zur qualitativen Modellierung sind die Petrinetze (PN). Diese wurden in den 1960er Jahren von Carl Adam Petri entwickelt, um nebenläufige Prozesse zu modellieren (Petri, 1962). PN sind bipartite Graphen, dessen Kanten gewichtet und gerichtet sind. Eine dieser Mengen repräsentiert die aktiven Teile des Modells, wie zum Beispiel Enzym-katalysierte Reaktionen, die andere Menge repräsentiert dessen passiven Teile, wie zum Beispiel Metaboliten. Die Knoten sind durch gerichtete und gewichtete Kanten verbunden. Auf den Plätzen können Marken platziert werden, welche die vorhandene Menge einer Substanz repräsentieren. Eine erste Anwendung von PN in der Systembiologie erfolgte durch Reddy et al. (1993) anhand eines metabolischen Modells des Fruktose-Metabolismus der Leber. Seitdem wurden PN auf weitere Bereiche der

Systembiologie angewendet, so zum Beispiel für genregulatorische Netzwerke (Chaouiya et al., 2008; Grunwald et al., 2008) oder Signaltransduktionswege (Sackmann et al., 2006; Grafahrend-Belau et al., 2008). Eine umfassende Übersicht über die verschiedensten Anwendungen von PN in der Systembiologie wird durch Pinney et al. (2003), Chaouiya (2007) und Koch et al. (2011) gegeben. Für die Analyse von PN stehen eine Vielzahl von Methoden zur Verfügung. So lässt sich auch hier die Analyse von Fließgleichgewichten anwenden, die T-Invarianten (Lautenbach, 1973; Murata, 1989). Basierend auf den T-Invarianten wurden weitere Methoden entwickelt, so unter anderem die *maximal common transition sets* (MCT-Sets), um kleinste funktionelle Einheiten innerhalb des PN zu identifizieren (Sackmann et al., 2006). Im PN können Elemente deaktiviert werden, um die Auswirkungen dieses Knock-outs untersuchen zu können (Grunwald et al., 2008). Durch die Marken und fest definierte Schaltregeln für das PN lässt sich das dynamische Verhalten eines PN untersuchen (Balazki et al., 2015).

Für den PN-Formalismus wurden viele Erweiterungen entwickelt, um komplexere Systeme modellieren zu können. In gefärbten PN (Jensen, 2013) können den Marken unterschiedliche Werte zugewiesen werden, genannt Farben. Auch mit diesen lässt sich zum Beispiel ein Signaltransduktionsweg modellieren (Lee et al., 2006). Daneben gibt es unter anderem auch stochastische, hybride, hierarchische oder zeitabhängige PN. Durch diese Erweiterungen kann die Modellierung eines PN qualitativ starten, aber Schritt für Schritt zu einem quantitativen Modell erweitert werden (Chen et al., 2011).

### 1.1.2 Quantitative Modellierung

Die quantitative Modellierung kann zum Einsatz kommen, wenn für ein System möglichst alle nötigen Daten, wie Reaktionsgeschwindigkeiten oder Konzentrationen, zur Verfügung stehen. Für die quantitative Modellierung kommen Differenzialgleichungen zum Einsatz, die Änderungsraten kontinuierlicher Variablen beschreiben und für deren Erstellung diese Daten benötigt werden. Mit Hilfe solcher Gleichungen können dynamische Systeme modelliert werden. Eine Methode hierzu sind gewöhnliche Differenzialgleichungssysteme (ODE), mit denen sich die Änderung der Substanzmengen im System in Abhängigkeit der Zeit beschreiben lassen. Anwendung findet eine solche Modellierung sowohl bei metabolischen Systemen (Chassagnole et al., 2002), Signaltransduktionswegen (Tyson et al., 2003) als auch bei Modellen der Genexpression (Chen et al., 1999). Für die Modellierung eines ODE-Modells wird jedoch exaktes Wissen über das zu modellierende System vorausgesetzt, um die entsprechenden Reaktionskonstanten und kinetischen Parameter richtig zu setzen. Dies verhindert oft die Modellierung größerer Modelle oder gar genomweiter Modelle. Mit einem solchen Gleichungssystem lässt sich das zeitliche Verhalten des Modells simulieren oder der Einfluss verschiedener Startbedingungen beobachten. Mit Hilfe der metabolischen Flussanalyse (Wiechert, 2001) existiert eine Methode, um den Stoffffluss in einem Modell zu untersuchen und eine Analyse des Fließgleichgewichts durchzuführen (Kruger et al., 2007; Ahn und Antoniewicz, 2011; Swarup et al., 2014). Um zu untersuchen,

wie sich beispielsweise die Ausbeute eines Metaboliten unter bestimmten Umständen maximieren lässt, kann eine Flussgleichgewichtsanalyse (Orth et al., 2010) durchgeführt werden (Grafahrend-Belau et al., 2009; Mahadevan et al., 2002; Poolman et al., 2009). Neben ODE-Systemen können auch stochastische Differenzialgleichungen (SDE) oder partielle Differentialgleichungen (PDE) verwendet werden, um stochastische Effekte, beziehungsweise räumliche Verteilungen mit in die Modellierung einzubeziehen (Turner et al., 2004).

## 1.2 Anwendungen zur Modellierung und Analyse biologischer Systeme

Für die quantitative Modellierung steht eine Vielzahl an Anwendungen bereit. CellDesigner (Funahashi et al., 2003), COPASI (Hoops et al., 2006) oder CellIllustrator (Nagasaki et al., 2010) sind nur einige davon. Eine umfassendere Auflistung ist in Koch et al. (2011) zu finden.

Für die qualitative Modellierung mit PN existiert ein reichhaltiges Angebot von Anwendungen, da dieses Konzept nicht nur in der Systembiologie Anwendung findet. Eine ausführliche Übersicht bietet die *Petri Net Tool Database* (Haustermann, 2016). Für die Anwendung VANTED (Rohn et al., 2012) existiert die Erweiterung *PetriNet* (Hartmann et al., 2012) zur Modellierung von PN. Diese ermöglicht neben der Modellierung eines PN auch dessen Simulation sowie die Berechnung der Invarianten und des Erreichbarkeitsgraphen. Eine in der Systembiologie häufig eingesetzte Anwendung ist Snoopy (Fieber, 2004). Der Fokus von Snoopy liegt auf der Bereitstellung möglichst vieler Varianten des PN-Formalismus. Mit Snoopy können 15 dieser Varianten modelliert werden, darunter zeitabhänige PN, gefärbte PN, kontinuierliche PN und stochastische PN. Neben der Modellierung ist eine Simulation des PN möglich. Für die verschiedenen Erweiterungen des PN-Formalismus existieren weitere spezialisierte Anwendungen. Für stochastische PN kann SPNP (Ciardo et al., 1989) oder DSPNexpress (Lindemann, 1995) verwendet werden. Neben stochastischen PN können mit GreatSPN (Chiola et al., 1995) auch zeitabhängige PN analysiert werden.

Die Modellierung Boolescher Netzwerke wird zum Beispiel durch BooleanNet (Albert et al., 2008), BoolNet (Müssel et al., 2010) oder das Cytoscape Plugin SimBoolNet (Zheng et al., 2010) ermöglicht. Eine ausführliche Übersicht über Anwendungen zur Modellierung boolescher Netze ist in Wang et al. (2012) zu finden. Die Modellierung von Bayesschen Netzen kann mit Anwendungen wie beispielsweise WinBUGS (Lunn et al., 2000) oder CellNetAnalyzer (Klamt et al., 2007) realisiert werden.

Anwendungen zur Visualisierung von biologischen Modellen stehen mit Cytoscape (Shannon et al., 2003), VANTED (Rohn et al., 2012) oder BioUML (Kolpakov, 2002) bereit. Cytoscape und VANTED bieten zudem die Möglichkeit an, experimentelle Daten in das Netzwerk zu integrieren und stellen Erweiterungen bereit, die den Funktionsumfang der Anwendung erweitern und an die jeweiligen Bedürfnisse des Modells anpassen können.

### 1.2.1 Austauschformate für biologische Modelle

Die große Zahl an Anwendungen zur Modellierung biologischer Modelle und Datenbanken hat es notwendig gemacht, einheitliche Dateiformate zu entwickeln. Mit Hilfe dieser lassen sich erstellte Modelle zwischen den Anwendungen und Datenbanken austauschen.

Die *Systems Biology Markup Language* (SBML) (Hucka et al., 2003; Finney und Hucka, 2003) ist ein freies und offenes XML-Format (Bray et al., 1998) und ein weit verbreiteter Standard in der Systembiologie. Es ermöglicht, die Abbildung von quantitativen und qualitativen Modellen und die Einteilung des Modells in Zellkompartimente. Durch verschiedene *Packages* können zudem weitere, spezifischere Informationen im Modell hinterlegt werden. So wurden *Packages* zum Modellieren räumlicher Vorgänge oder dynamischer Prozesse entwickelt. Eine Übersicht über alle *Packages* ist im Internet (SBML-Packages, 2016) verfügbar. Für die Integration von SBML in neue Software stehen zwei offizielle Bibliotheken zur Verfügung, libSBML (Bornstein et al., 2008) und jSBML (Rodriguez et al., 2015). Aktuell unterstützen mehr als 281 Anwendungen das SBML-Format. Eine Übersicht über diese gibt der *SBML Software Guide* (SBML-Software-Guide, 2016). Unterstützt eine Anwendung den Import einer SBML Datei, so bedeutet dies jedoch nicht automatisch, dass auch ein Export in das SBML-Format angeboten wird. Datenbanken, wie Reactome (Milacic et al., 2012; Croft et al., 2014) oder Biomodels (Li et al., 2010), unterstützen ebenfalls Modelle dieses Formats.

Im BioPAX-Format (Demir et al., 2010) werden die Modelle mit Hilfe der *Web Ontology Language* (McGuinness und van Harmelen, 2004) beschrieben und mit der RDF/XML-Syntax (Beckett und McBride, 2004) gespeichert. BioPAX kann sowohl Signaltransduktionswege, molekulare Interaktionen als auch genregulatorische Netzwerke abbilden. Datenbanken wir BioCyc (Caspi et al., 2008) und Reactome (Milacic et al., 2012; Croft et al., 2014) oder Anwendungen wie Cytoscape (Shannon et al., 2003) unterstützen dieses Format. Mit *Paxatools* (Demir et al., 2013) wird eine Java Bibliothek zum Verwenden des BioPAX Formats bereitgestellt.

Mit der *Systems Biology Graphical Notation* (SBGN) existiert ein offenes XML-Format, mit dem eine standardisierte grafische Notation von biologischen Prozessen ermöglicht wird (Novere et al., 2009). Dieses Format wird schon von zahlreichen Anwendungen, unter anderem CellDesigner (Funahashi et al., 2003), und Datenbanken, beispielsweise Biomodels (Li et al., 2010), unterstützt.

Neben diesen Formaten existieren noch weitere spezifischere Formate. Zum Austausch von PN ist hier zum Beispiel die *Petri Net Markup Language* (Weber und Kindler, 2003) oder das PNT-Format (PNT, 2016) zu nennen. Eine Repräsentation der Modelle als Graph kann im GraphML Format (Brandes et al., 2013) vorgenommen werden, einem allgemeinen Austauschformat für Graphen.

## 1.3 Motivation

Die Modellierung biologischer Systeme ist ein wichtiger Bestandteil der Systembiologie. Durch Hochdurchsatz-Methoden und die Erschließung der *-om*-Daten werden immer mehr Informationen für eine solche Modellierung zugänglich. Diese Daten sind jedoch in den meisten Fällen komplex, d.h., dass beispielsweise nicht alle Reaktionskonstanten für alle Reaktionen eines Modells vorhanden sind. Dies erlaubt keine exakte quantitative Modellierung, weshalb in vielen Fällen auf eine qualitative Modellierung zurückgegriffen werden muss. Für eine solche Modellierung bietet sich das Konzept der Petrinetze (PN) an. Es wird seit den 1990er Jahren in der Systembiologie eingesetzt, daher existieren viele Modelle auf dessen Basis und PN-Modelle können durch verschiedenste Erweiterungen in quantitative Modelle überführt werden. Das Formulieren aussagenlogischer Funktionen, wie bei der booleschen Modellierung, oder die Definition von Zufallsvariablen im Falle der Bayesschen Netze entfällt bei den PN. Für PN wurden im Laufe der Jahre viele, auch systembiologisch motivierte, Analysemethoden entwickelt, die eine Analyse und biologische Interpretation deren Resultate zulassen. Die Visualisierung von PN ist zudem intuitiv und einfach zu handhaben.

Viele vorhandene Anwendungen zur Modellierung und Analyse von PN stammen nicht aus der Systembiologie, ihre Terminologie und Funktionen sind nicht auf diese angepasst. Die systembiologisch motivierten Analysemethoden, wie beispielsweise die Knockout-Analyse oder die MCT-Sets, sind oft auch nicht in die vorhandenen Anwendungen integriert. Daher war die Motivation dieser Arbeit, eine Anwendung zu entwickeln, welche die PN-Modellierung und Visualisierung für die Systembiologie ermöglicht und eine möglichst große Zahl von Analysemethoden bereit stellt. In dieser Anwendung sollte eine möglichst große Auswahl an Analysemethoden zusammenfasst und neben der strukturellen Analyse von PN auch die dynamische Untersuchung von PN ermöglicht sowie neben der qualitativen Modellierung auch die Möglichkeit zur quantitativen Analyse bieten. Eine Anwendung, die all diese Analysemethoden vereint, existiert im systembiologischen Kontext noch nicht. Zunächst soll dies für einfache PN-Systeme ermöglicht werden, die Unterstützung von Erweiterungen sollte zunächst nicht im Fokus stehen. Mit der immer größer werdenden Menge an Wissen steigt auch die Komplexität der Modelle. Mit der zunehmenden Komplexität steigt jedoch auch die Menge der Resultate der Analysemethoden für PN, wie zum Beispiel den T-Invarianten (Klamt und Stelling, 2002). Dies wiederum erschwert die Auswertung und Interpretation der Resultate. Daher ist eine weitere große Motivation dieser Arbeit, dass die Anwendung den Modellierer bei diesem Prozess bestmöglichst unterstützt, so etwa durch die Visualisierung der Resultate auf dem PN oder weitere andere Arten der grafischen Repräsentation dieser, wie zum Beispiel Resultate von Clusterverfahren als Bäume. Dies soll für jede der bereitgestellten Analysemethoden möglich sein. Die Anwendung soll weiter eine breite Palette von existierenden Austauschformaten unterstützen. Diese sollen, soweit möglich, nicht nur gelesen sondern auch geschrieben werden können. So können die

erstellten Modelle leicht mit anderen Anwendungen oder Modellierern ausgetauscht werden oder in Datenbanken hinterlegt werden. Die Anwendung soll so konzipiert sein, dass die Einbindung neuer Analysemethoden leicht und unkompliziert vorgenommen werden kann. Durch eine quelloffene Entwicklung und Veröffentlichung soll dies auch für Dritte möglich sein.

## 1.4 Aufbau der Arbeit

Die Arbeit wird die Implementation und die Funktionalitäten der Software MonaLisa beschreiben, mit der PN-Modelle biologischer Systeme modelliert und analysiert werden können. Im ersten Teil des folgenden Kapitels wird die Implementation einiger essentieller Klassen der Software beschrieben und diskutiert. Darauf folgt eine Betrachtung der wichtigsten Funktionalitäten, die die Modellierung des PN und der verfügbaren Analysemethoden betreffen. Im Anschluss an dieses Kapitel folgt eine kurze Zusammenfassung.

# Kapitel 2

# Material und Methoden

## 2.1 Der Petrinetz-Formalismus

Petrinetze (PN) wurden 1962 von Carl Adam Petri entwickelt, um nebenfläufige Schaltprozesse zu beschreiben (Petri, 1962). In der Biologie wurden Petrinetze das erste Mal Anfang der 1990er Jahre angewandt, um biochemische Systeme zu beschreiben (Reddy et al., 1993). Einen Überblick über die verschiedenen Anwendungsmöglichkeiten von Petrinetzen in der Systembiologie liefern Koch et al. (2011) und Chaouiya (2007).

Ein PN besteht aus den zwei disjunkten Knotenmengen $P$ und $T$, genannt *Plätze* und *Transitionen*, wobei $P \cap T = \emptyset$ gilt. Knoten aus diesen Mengen sind durch gerichtete und gewichtete Kanten verbunden, wobei keine Kanten zwischen Knoten aus derselben Knotenmenge existiert. Die Menge aller gerichteten und gewichteten Kanten wird durch die Menge $E = (P \times T) \cup (T \times P)$ repräsentiert. Die Funktion $w : E \to \mathbb{N}_0$ weist jeder Kante $e \in E$ ein Kantengewicht zu. Somit kann die Topologie eines PN durch das Quadruple $PN = (P, T, F, w)$ beschrieben werden.

Die direkten Vorgängerknoten einer Transition werden *Vorplätze* genannt, die eines Platzes *Vortransitionen*. Analog dazu werden die direkten Nachfolgeknoten einer Transition *Nachplätze* genannt, die eines Platzes *Nachtransitionen*. Die Definition dieser Mengen ist gegeben durch:

- die Menge der Vorplätze $\bullet t = \{p \in P \mid (p, t) \in E\}$,
- die Menge der Nachplätze $t\bullet = \{p \in P \mid (t, p) \in E\}$,
- die Menge der Vortransitionen $\bullet p = \{t \in T \mid (t, p) \in E\}$,
- die Menge der Nachtransitionen $p\bullet = \{t \in T \mid (p, t) \in E\}$.

Die Kardinalität der Mengen $\bullet t$, dem eingehenden Knotengrad, und $t\bullet$, dem ausgehenden Knotengrad, wird als Knotengrad $k$ einer Transition $t$ bezeichnet, es gilt also $k(t) = | \bullet t | + | t \bullet |$. Diese Definition gilt analog auch für Plätze, also $k(p) = | \bullet p | + | p \bullet |$.

Das dynamische Verhalten eines PN wird durch die Einführung diskreter Einheiten, der *Marken*, möglich. Diese Marken werden auf allen Plätzen $p \in P$ platziert, die daraus

resultierende Verteilung wird *Markierung* genannt. Zu einem bestimmten Zustand $k$ des PN liefert die Funktion $m_k : P \to \mathbb{N}_0$ für jeden Platz $p \in P$ die Anzahl der sich auf diesem Platz befindlichen Marken. Die Startmarkierung, also die Verteilung der Marken zu Beginn der Untersuchung des dynamischen Verhaltens, wird $m_0$ genannt. Durch die Erweiterung zu dem Quintupel $PN = (P, T, E, w, m_0)$ wird ein dynamisches Verhalten eines PN definiert.

Die Simulation dieses dynamischen Verhaltens folgt festen Regeln, den *Feuerregeln.* Das *Feuern* findet hierbei an den Transitionen statt und bewegt die Marken zwischen Plätzen. Eine Transition $t_i \in T$ kann nur feuern, wenn sie *aktiviert* ist, also auf jedem Vorplatz $p_i \in {}^\bullet t$ die Anzahl der Marken mindestens so groß ist wie das Kantengewicht $w(p_i, t_i)$ der verbindenden Kante, also $\forall p_i \in {}^\bullet t_i : m(p_i) \geq w(p_i, t_i)$ gilt. Eine Transition $t_i \in T$, für welche ${}^\bullet t_i = \emptyset$ gilt, ist immer aktiviert und wird *Input-Transition* genannt. Solch eine Transition führt neue Marken in das PN ein. Eine Transition $t_i \in T$, für welche gilt $t_i{}^\bullet = \emptyset$, wird *Output-Transition* genannt. Eine solche unterliegt den oben beschriebenen Regeln zur Aktivierung, entfernt beim Feuern jedoch Marken aus dem PN.

Beim Feuern einer Transition $t_i \in T$ werden die dem Kantengewicht entsprechende Anzahl an Marken von den Vorplätzen entfernt. Zeitgleich wird auf allen Nachplätzen von $t_i$ eine Anzahl von Marken neu erzeugt, die dem Gewicht der entsprechenden Kante entspricht. Eine Definition dieser Schritte wird gegeben durch:

– Entfernt der Marken: $\forall p_i \in {}^\bullet t_i : m_{k+1}(p_i) = m_k(p_i) - w(p_i, t_i)$

– Erzeugen der Marken: $\forall p_i \in t_i{}^\bullet : m_{k+1}(p_i) = m_k(p_i) + w(p_i, t_i)$

Im Falle eines metabolischen Systems repräsentieren Plätze die Metaboliten und Transitionen die Reaktionen dieses Systems. Die Kantengewichte entsprechen den stöchiometrischen Faktoren einer Reaktion. Die Marken spiegeln die Anzahl der Moleküle eines Metaboliten wider, zum Beispiel als genaue Molekülanzahl oder als $Mol$. Das Feuern einer Transition entspricht dem Stattfinden einer Reaktion. Hierbei ist zu beachten, dass im hier beschriebenen PN das Feuern einer Transition keine Zeit benötigt, was bei einer biochemischen Reaktion nicht der Fall ist. Ein PN mit diesen Feuerregeln wird ein *P/T-System* genannt. Im weiteren Text verweist die Abkürzung PN auf ein solches P/T-System.

Abbildung 2.1 zeigt ein Beispiel für ein PN. Plätze werden hier durch Kreise repräsentiert und Transitionen durch Rechtecke. Die Kantengewichte werden durch Zahlen an den Kanten angegeben, wobei Kanten mit einem Gewicht von 1 keine Beschriftung besitzen. Punkte auf den Plätzen repräsentieren die Marken des PN.

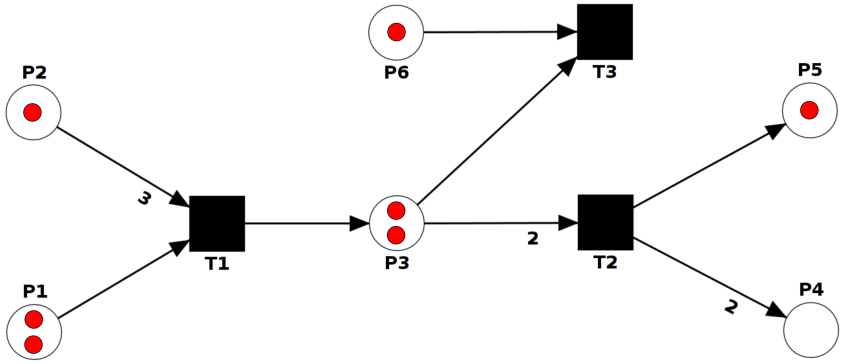


**Abbildung 2.1:** Die Abbildung zeigt ein kleines Beispiel für die grafische Repräsentation eines PN. Die Kreise stehen für die Plätze des PN, die Rechtecke für die Transitionen. Kantengewichte werden durch Beschriftung der Kanten angegeben, wobei eine Kante mit Gewicht 1 keine solche erhält. Marken werden durch Punkte auf den Plätzen repräsentiert. Die Transitionen *T2* und *T3* sind aktiviert, da auf ihren Vorplätzen *P3* und *P6* genügend Marken vorhanden sind. Wohingegen Transition *T1* nicht aktiviert ist, da auf *P2* zwei Marken zu wenig vorhanden sind.

### 2.1.1 Elementare Netzeigenschaften von Petrinetzen

Die in den folgenden Abschnitten vorgenommenen Definitionen folgen Starke (1990).

#### Gewöhnlichkeit

Ein PN wird gewöhnlich (*ordinary*) genannt, falls alle Kanten des PN ein Gewicht von 1 haben.

#### Homogenität

Wenn für jeden Platz eines PN die dort startenden Kanten das gleiche Gewicht haben, so wird dieses homogen (*homogenous*) genannt.

#### Nichtblockierende Vielfachheit

Wenn für jeden Platz eines PN gilt, dass das minimale Gewicht der eintreffenden Kanten nicht kleiner ist als das maximale Gewicht der ausgehenden Kanten, so wird dies als nichtblockierende Vielfachheit (*non-blocking multiplicity*) bezeichnet.

#### Reinheit

Ein PN ist rein (*pure*), falls für keine Transition einer deren Vorplätze zugleich ein Nachplatz dieser ist.

#### Konservativität und Sub-Konservativität

Ein PN heißt konservativ (*conservative*), wenn für alle Transitionen die Summe der Kantengewichte aller eingehenden Kanten der Summe der Kantengewichte aller ausgehenden

Kanten entspricht. In einem solchen PN ändert sich die Gesamtzahl aller Marken nicht, da beim Feuern einer Transition immer genau so viel Marken erzeugt werden, wie vernichtet werden.

Falls für alle Transitionen gilt, dass die Summe der Kantengewichte aller Nachplätze höchstens so hoch ist wie die Summe der Kantengewichte aller Vorplätze, so ist das PN sub-konservativ. In diesem Fall bleibt die Gesamtzahl aller Marken nicht konstant, kann sich aber nicht erhöhen.

#### Statische Konfliktfreiheit

Ein statischer Konflikt um Marken eines Platzes entsteht, wenn zwei Transitionen diesen Platz als gemeinsamen Vorplatz haben. Ist dies für keinen Platz der Fall, ist das PN statisch konfliktfrei (*static conflict free*).

#### Zusammenhang

Wenn von jedem Knoten eines PN ein ungerichteter Weg zu jedem anderen Knoten existiert, so wird das PN zusammenhängend (*connected*) genannt. Die Richtung aller Kanten wird für diese Untersuchung ignoriert.

#### Starker Zusammenhang

Wenn ein PN zusammenhängend ist, kann zusätzlich überprüft werden, ob ein gerichteter Weg zwischen allen Knotenpaaren existiert. Ist dies der Fall, so wird das PN als stark zusammenhängend (*strongly connected*) bezeichnet.

### 2.1.2 Transitions-Invarianten

Das Konzept der Transitions-Invarianten (T-Invarianten) ist ausgiebig in der Literatur beschrieben (Lautenbach, 1973; Murata, 1989). Die Definition einer T-Invariante erfordert zunächst die Einführung der *Inzidenzmatrix*. Die Inzidenzmatrix $C$ eines PN ist eine $m \times n$ Matrix, wobei $m$ der Anzahl an Plätzen und $n$ der Anzahl an Transitionen des PN entspricht. Ein Eintrag $c_{i,j}$ entspricht der Änderung der Anzahl von Marken auf dem Platz $p_i \in P$, wenn die Transition $t_j \in T$ einmal feuert:

$$c_{i,j} := \begin{cases} w(t_j, p_i) & , falls\ (t_j, p_i) \in E \\ -w(p_i, t_j) & , falls\ (p_i, t_j) \in E \\ w(t_j, p_i) - w(p_i, t_j) & , falls\ (t_j, p_i) \in E \wedge (p_i, t_j) \in E \\ 0 & , sonst. \end{cases} \tag{2.1}$$

Gegeben sei eine Sequenz von Transitionen $s = (t_1, ..., t_i, ...t_n)$ und ein dazugehöriger Vektor $x = (x_1, ..., x_i, ...x_n)$ mit $x_i \in \mathbb{N}$, wobei $x_i$ die Anzahl der Feuerungen von $t_i$ in $s$ angibt. Der Vektor $x$ wird auch Parikh-Vektor genannt. So ist eine Veränderung der

Verteilung der Marken durch $\Delta m = Cx$ definiert. Der Vektor $\Delta m$ beinhaltet an jeder Stelle $i$ die Veränderung an Marken auf dem Platz $p_i$. Der Vektor $supp(x) = (k_1, ..., k_i, ..., k_n)$ mit $k_i = 1$, falls $x_i > 0$, ansonsten $k_i = 0$, wird der *Support* des Vektors $x$ genannt. Hierbei gibt $|supp(x)|$ die Anzahl der Einträge in $x$ an, für welche $x_i > 0$ gilt.

Eine T-Invariante ist eine Menge von Transitionen, welche durch eine bestimmte Anzahl von Feuerungen die Markierung eines PN nicht ändern, also $\Delta m = 0$ zutrifft. Diese Menge von Transitionen bildet, zusammen mit ihren Vor- und Nachplätzen, ein zusammenhängendes Subnetz. Daraus ergibt sich, dass eine T-Invariante die Lösung der Gleichung

$$Cx = 0 \tag{2.2}$$

ist. Die triviale Lösung dieser Gleichung ist $\forall i : x_i = 0$, alle andere Lösungen werden *semi-positiv* genannt. Alle Vektoren $x$, welche die Gleichung 2.2 lösen und keine triviale Lösung darstellen, bilden die T-Invarianten-Menge $J = \{\ x \mid x_i \geq 0 \wedge\ |supp(x)| > 0\ \}$. Eine andere Darstellungsmöglichkeit ist eine Matrix $V = n \times |J|$, wobei jede Zeile eine Transition repräsentiert und jede Spalte eine T-Invariante. Dabei steht $v_{i,j}$ für das Auftreten der Transition $i$ in der T-Invariante $j$. Wenn mindestens ein Vektor $x$ existiert, welcher Gleichung 2.2 löst, steigt die Anzahl aller möglichen Lösungen ins Unendliche, da jeder Vektor $\alpha x,\ \alpha \in \mathbb{N}$ ebenfalls eine Lösung ist. Daher ist es nötig, die T-Invarianten-Menge auf *minimale semi-postivie T-Invarianten* zu beschränken.

**Definition 2.1** *Eine Invarianten-Menge von ganzzahligen Lösungen $x$, der Gleichung 2.2, wird minimal ($J_{min}$) genannt, falls*

- $\nexists q \in J : \exists w \in J : supp(q) \subset supp(w)$
- $\nexists \alpha \in \mathbb{N}_{>1} : \forall x \in J : x_i = \alpha x'_i,\ x'_i \in \mathbb{N}$

*zutrifft.*

Falls jede Transition eines PN an mindestens einer T-Invariante beteiligt ist, ist das PN von T-Invarianten überdeckt, CTI (*coverd by t-invariants*), wobei diese T-Invarianten nicht minimal sein müssen.

**Definition 2.2** *Ein Petrinetz $N = (P, T; E, w, m_0)$ mit der T-Invarianten-Menge $J$ hat die* CTI *Eigenschaft, wenn $\forall t_i \in T : \exists x \in I : x_i > 0$ zutrifft.*

Der Begriff der minimalen semi-positiven T-Invarianten ist äquivalent zum Konzept der Elementarmoden (*Elementary Modes*) (Schuster und Hilgetag, 1994; Koch et al., 2005). Ein Algorithmus zur Berechnung von T-Invarianten wird durch Koch und Ackermann (2013) beschrieben.

Im weiterem Text bezieht sich der Begriff *T-Invarianten* auf die T-Invarianten-Menge $J_{min}$ und der Begriff *T-Invariante* auf ein $x \in J_{min}$. Der Begriff einer trivialen T-Invariante

verweist im weiteren Text auf eine T-Invariante, welche nur aus maximal zwei Transitionen besteht, also beispielsweise die Hin- und Rückreaktion zwischen denselben Stoffen modelliert.

Im Falle eines metabolischen PN repräsentiert eine T-Invariante eine Menge von Enzymen, welche sich im Fließgleichgewicht befinden. Ein Beispiel einer T-Invariante wird in Abbildung 2.2 gezeigt.

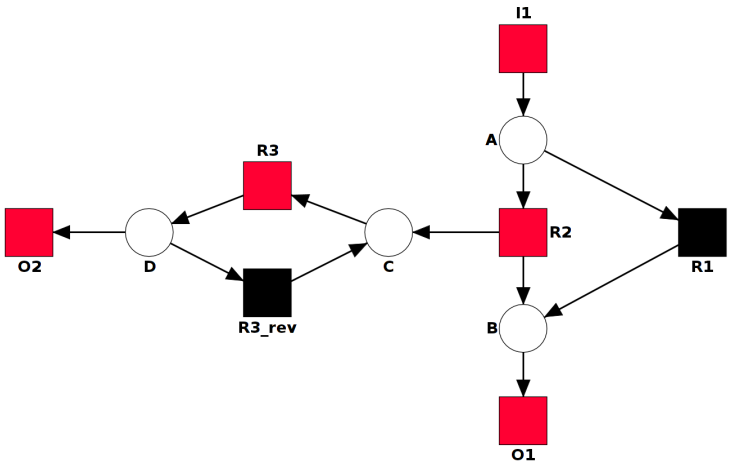


**Abbildung 2.2:** Die Abbildung zeigt ein PN und eine eingefärbte (minimale, semi-positive) T-Invariante des PN. Wenn jede der rot markierten Transitionen (*I1, R2, O1, R3, O2*) einmal gefeuert hat, ist die Verteilung der Marken im PN wie zuvor. Eine weitere T-Invariante sind die Transitionen (*I1, R1, O1*).

### 2.1.3 Platz-Invarianten

Die Definition von Platz-Invarianten (P-Invarianten) erfolgt analog zu der Definition der T-Invarianten. Zu lösen ist hier die Gleichung

$$C^T x = 0 \tag{2.3}$$

Eine P-Invariante stellt eine Menge von Plätzen dar, deren Gesamtmenge an Marken konstant sind, egal in welcher Markierung $m$ sich das PN befindet. Im Falle eines metabolischen PN repräsentiert eine P-Invariante eine Konservierung von Substanzen, da sich ihre Menge nicht ändert. Ein solches Beispiel ist in Abbildung 2.3 zu finden, in der ein Modell der Michaelis-Menten-Theorie (Michaelis und Menten, 1913) gezeigt wird. Hier ist die Menge an Substrat und dem Substrat-Enzym-Komplex konserviert.

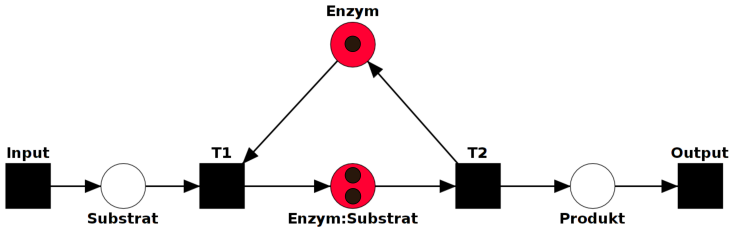


**Abbildung 2.3:** Die Abbildung zeigt ein PN und eine eingefärbte P-Invariante des PN. Es wird ein Modell der Michaelis-Menten-Theorie gezeigt, also die Beziehung von Substrat, Enzym und dem Produkt der enzymatischen Reaktion. Die Anzahl der Marken auf den Plätzen (*Enzym, Enzym:Substrat*) ist, egal in welchem Zustand sich das PN befindet, konstant. Wird eine Marke von *Enzym* konsumiert, so wird diese immer auf dem Platz *Enzym:Substrat* erzeugt und andersherum.

### 2.1.4 *Maximal Common Transition Sets*

Die *maximal common transition sets* (MCT-Sets) wurden entwickelt, um die Menge an T-Invarianten zu reduzieren und im PN kleinste funktionale Einheiten zu finden (Sackmann et al., 2006). Gegeben sei das MCT-Sets mit den Transitionen $\{t_1, ..., t_i, ..., t_j\}$. Die Transitionen $t_i$ und $t_j$ werden in einem MCT-Sets-Sets zusammengeführt, wenn sie in den exakt gleichen T-Invarianten vorhanden sind. Das heißt, falls $t_i$ in einer T-Invariante enthalten ist, muss $t_j$ ebenfalls vorhanden sein, die Abwesenheit von $t_i$ bedingt auch die Abwesenheit von $t_j$. Eine Menge von Transitionen $A \subseteq T$ ist ein MCT-Sets, falls gilt:

$$\forall i \in I_{min} : A \subseteq supp(i) \vee A \cap supp(i) = \emptyset \tag{2.4}$$

Ein MCT-Sets bildet ein disjunktes Subnetz des PN. Gilt für ein PN die CTI Eigenschaft nicht, so bilden alle Transitionen, welche in keiner T-Invariante enthalten sind, ein eigenes MCT-Sets. Anstelle des Supportvektors der T-Invarianten, können diese auch auf deren Parikh-Vektoren berechnet werden.

### 2.1.5 T-Cluster

Die MCT-Sets stellen ein striktes Kriterium dar, um funktionale Einheiten im PN zu finden. Die T-Cluster wurden entwickelt, um Überlappungen der T-Invarianten bei der Suche nach funktionalen Modulen im PN zuzulassen (Grafahrend-Belau et al., 2008). Die T-Invarianten werden hierbei benutzt, um mit Hilfe eines Distanzmaßes und einer hierarchischen Clustermethode dieses Ziel zu erreichen. Die Distanz zweier T-Invarianten wird auf Grundlage ihrer Supportvektoren berechnet. Eine große Distanz zwischen zwei T-Invarianten bedeutet also, dass sie sehr unterschiedliche Transitionen enthalten, wohingegen eine kleine Distanz darauf schließen lässt, dass beide T-Invarianten eine große Schnittmenge an Transitionen haben. Das Ergebnis dieser Methode ist eine Cluster-Hierarchie der T-Invarianten, welche eine Einteilung in Module zulässt. Die in dieser Arbeit verwendeten Distanzmaße sind in Abschnitt 2.3 beschrieben, die verwendeten Clustermethoden in Abschnitt 2.4.

### 2.1.6 Knock-out-Analyse

Die Knock-out-Analyse zeigt auf, welcher Teil eines PN aktiv bleibt und welcher Teil seine biologische Bedeutung verliert, wenn Teile des PN gelöscht werden (Grunwald et al., 2008). Hierzu werden die Transitionen oder Plätze, deren Augenmerk der Knock-out-Analyse gilt, aus dem PN gelöscht und anschließend die T-Invarianten neu berechnet. Diese neu berechneten T-Invarianten werden nun mit den T-Invarianten des originalen PN verglichen. Die Teile, welche nicht mehr von T-Invarianten überdeckt werden, haben ihre biologische Bedeutung verloren, da hier ein Fließgleichgewicht nicht mehr möglich ist. Jene Teile, die noch immer von T-Invarianten überdeckt sind, haben ihre biologische Bedeutung behalten und sind nicht von dem Knock-out betroffen.

### 2.1.7 *Minimal Cut Sets*

Das Ziel der *minimal cut sets* (MCS) ist ebenfalls die Untersuchung der Auswirkungen im Falle eines Wegfallens bestimmter Transitionen (Klamt und Gilles, 2004). Die genaue Fragestellung hier ist, welche anderen Transitionen inaktiviert werden müssen, damit eine ausgewählte Transition ebenfalls inaktiv wird, also nicht mehr von T-Invarianten überdeckt ist. Ein MCS ist eine Menge von Transitionen, deren Inaktivierung (Löschen) zu der Inaktivierung einer ausgewählten Transition führt. Ein Algorithmus zur Berechnung der MCS wird in Klamt und Gilles (2004) beschrieben.

## 2.2 Stochastische Methoden zur Simulation chemischer Systeme

Damit eine Reaktion stattfinden kann, müssen sich die daran beteiligten Moleküle im Raum treffen. Die Wahrscheinlichkeit, dass sich diese Moleküle treffen, steigt, je größer die Anzahl der Moleküle ist. Dies ist die Grundidee der von Gillespie entwickelten Methode zur stochastischen Simulation gekoppelter chemischer Systeme (Gillespie, 1977). Dazu muss die experimentell bestimmte Reaktionskonstante $k$ einer chemischen Reaktion in einem ersten Schritt in eine stochastische Reaktionskonstante $c$ umgerechnet werden. Die Formel hierzu hängt vom Grad der Reaktion ab:

- $c = k\ V\ N_A$ für Reaktionen nullter Ordnung,
- $c = k$ für Reaktionen erster Ordnung der Form $A \rightarrow B$,
- $c = \frac{k}{V\ N_A}$ für Reaktionen zweiter Ordnung der Form $A + B \rightarrow C$ und
- $c = \frac{2k}{V\ N_A}$ für Reaktionen zweiter Ordnung der Form $2A \rightarrow B$.

Hierbei ist $V$ das Reaktionsvolumen und $N_A = 6 \cdot 10^{23}$ die Avogadro-Konstante. Eine allgemeine Formel für die stochastische Reaktionskonstante der Reaktion ist:

$$c(t) = \frac{k(t)\ \prod_{p_j \in {}^\bullet t} w(p_j, t)!}{(V\ N_A)^{(x-1)}}. \tag{2.5}$$

Die Reaktionsrate $a$ einer Reaktion setzt sich aus der stochastischen Reaktionskonstante und einem Faktor $h$ zusammen:

$$a = c\ h. \tag{2.6}$$

Der Faktor $h(t)$ steht für die Anzahl verschiedener Kombinationen, die alle vorhandenen Moleküle der Ausgangsstoffe einer Reaktion miteinander eingehen können. Die Formel zur Berechnung dieser hängt von der Ordnung der Reaktion ab:

- $h = 1$ für Reaktionen nullter Ordnung,
- $h = X_A$ für Reaktionen erster Ordnung der Form $A \rightarrow B$,
- $h = X_A X_B$ für Reaktionen zweiter Ordnung der Form $A + B \rightarrow C$ und
- $h = \frac{1}{2} X_A (X_A - 1)$ für Reaktionen zweiter Ordnung der Form $2A \rightarrow B$.

Hierbei steht $X_A$ für die Anzahl der Moleküle des Stoffes $A$. Eine nähere Beschreibung zur Berechnung der Reaktionsrate ist in Wilkinson (2011) zu finden.

Es existieren mehrere Varianten, um den Algorithmus von Gillespie zu implementieren. Im Folgenden werden zwei davon, der exakte Ansatz und eine approximative Methode, näher beschrieben.

### 2.2.1 Exakter stochastischer Simulationsalgorithmus

Der exakte Algorithmus besteht aus folgenden Schritten:

i Die Reaktionsrate $a(t)$ jeder Transition wird berechnet und $a^*$ als Summe aller Reaktionsraten gebildet.

ii Eine Zufallszahl $\tau$ aus einer gleichmäßigen Verteilung im Intervall $[0, 1]$ wird bestimmt. Diese wird verwendet, um die Zeitdifferenz zur aktuellen Simulationszeit $dt$ der nächsten Feuerung einer Transition zu bestimmen:

$$dt = -\frac{ln(1 - \tau)}{a^*} \tag{2.7}$$

iii Eine zweite Zufallszahl $\mu$ aus einer gleichmäßigen Verteilung im Intervall $[0, 1]$ wird bestimmt. Eine Transition $i$ wird ausgewählt, sodass die folgende Gleichung erfüllt wird:

$$\sum_{j=1}^{i-1} a(t_j) < \mu\ a^* \leq \sum_{j=1}^{i} a(t_j). \tag{2.8}$$

iv Transition $i$ wird gefeuert, die Anzahl der Marken auf den Plätzen aktualisiert und die Simulationszeit um $dt$ erhöht.

### 2.2.2 Approximativer stochastischer Simulationsalgorithmus

Die approximative Variante dieses Algorithmus dient zur Beschleunigung der Laufzeit und wurde von Gillespie und Petzold entwickelt (Gillespie und Petzold, 2003; Cao et al., 2005). Zum Durchführen dieses Algorithmus ist eine Definition *kritischer* Reaktionen nötig. Als solche werden Reaktionen bezeichnet, welche im aktuellen Zustand des chemischen Systems nicht mehr als 20 Mal stattfinden können. Im Kontext eines PN sind es also solche Transitionen, auf deren Vorplätzen nicht genügend Marken vorhanden sind, damit diese mehr als 20 Mal feuern können. Im Detail läuft ein Simulationsschritt wie folgt ab:

i Alle kritischen Transitionen werden bestimmt.

ii Zwei Feuerungszeiten $\tau_1$ für die nicht kritischen Transitionen und $\tau_2$ für die kritischen Transitionen werden bestimmt. Zur Bestimmung der Zeit $\tau_1$ existieren verschiedenste Ansätze (Gillespie und Petzold, 2003; Cao et al., 2006; Cao, 2010). Die Zeit $\tau_2$ wird wie im exakten Algorithmus, basierend allein auf den kritischen Reaktionen, berechnet.

iii **Im Falle $\boldsymbol{\tau_1 < \tau_2}$** wird keine kritische Transition gefeuert. Für alle anderen Transitionen wird eine Zufallszahl aus einer zufälligen Poisson-Variablen mit dem Mittelwert $a\ \tau_1$ bestimmt, und entsprechend oft gefeuert.
**Im Falle $\boldsymbol{\tau_2 < \tau_1}$** wird eine kritische Transition gemäß dem exakten Algorithmus gewählt und gefeuert. Alle anderen Transitionen werden hier wie im ersten Fall behandelt.

## 2.3 Distanzmaße

Um die Ähnlichkeit zwischen den zwei $p$-0dimensionalen Vektoren $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$ und $x_j = (x_{j1}, x_{j2}, ...., x_{jp})$ auszudrücken, wurden verschiedenste Distanzmaße entwickelt. Im Folgenden werden drei dieser Maße näher beschrieben, welche in dieser Arbeit verwendet werden. Zur Beschreibung der Distanzmaße werden folgende Definitionen benötigt:

$$\begin{aligned} n_{00} &= \sum_{k=1}^{p} I(x_{ik} = 0, x_{jk} = 0), \\ n_{01} &= \sum_{k=1}^{p} I(x_{ik} = 0, x_{jk} > 0), \\ n_{10} &= \sum_{k=1}^{p} I(x_{ik} > 0, x_{jk} = 0), \\ n_{11} &= \sum_{k=1}^{p} I(x_{ik} > 0, x_{jk} > 0). \end{aligned} \tag{2.9}$$

Die Funktion $I$ gibt 1 zurück, wenn alle Bedingungen erfüllt sind, ansonsten 0.

### 2.3.1 Tanimoto-Index

Der Tanimoto-Index berechnet die Ähnlichkeit zweier Vektoren als relativer Anteil gemeinsam vorhandener Merkmale. Basierend auf den vorhergehenden Definitionen ist der Tanimoto-Index zweier Vektoren $u$ und $v$ gegeben durch:

$$S_T(u,v) = \frac{n_{01} + n_{10}}{n_{01} + n_{10} + n_{11}}. \tag{2.10}$$

### 2.3.2 M-Koeffizient

Der M-Koeffizient, auch *Simple Matching* genannt, ist die Relation von gemeinsamen Werten zweier Vektoren in Bezug zur Länge der Vektoren. Der M-Koeffizient der beiden Vektoren $u$ und $v$ wird gegeben durch:

$$S_M(u,v) = \frac{n_{00} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}}. \tag{2.11}$$

### 2.3.3 Summe der absoluten Differenzen

Die Summe der absoluten Differenzen, auch als *Sum of Differences* bezeichnet, ist die aufsummierte (absolute) Differenz der entsprechenden Einträge zweier Vektoren und wird gegeben durch:

$$S_S(u,v) = \sum_{i=1}^{N} |u_i - v_i|. \tag{2.12}$$

## 2.4 Clustermethoden

Die im vorhergehenden Kapitel beschriebenen Distanzmaße können zum Beispiel dafür verwendet werden, um eine Menge von Vektoren mit hierarchischen agglomerativen Clustermethoden nach ihrer Ähnlichkeit zu sortieren und zu gruppieren. Im Folgenden werden die Methoden näher beschrieben, welche im Rahmen der Berechnung der T-Cluster in dieser Arbeit verwendet wurden. Basis dieser Clustermethoden ist eine Distanzmatrix. Diese repräsentiert den Abstand zwischen allen gegeben Vektoren, wobei die Einträge der Diagonalen auf 0 gesetzt sind. Eine Distanzmatix kann mit verschiedenen Methoden erstellt werden, von denen eine Auswahl im vorherigen Kapitel beschrieben wurde. Solch eine Distanzmatrix wird nun verwendet, um die beiden Elemente mit der geringsten Distanz zu ermitteln und zu einem neuen Cluster zu vereinen. Anschließend müssen die neuen Distanzen zwischen diesem Cluster und allen anderen Einträgen neu berechnet werden. Die Art und Weise der Berechnung dieser Distanzen ist der Unterschied der einzelnen Clustermethoden. Zu Beginn werden meist einzelne Elemente zu neuen Clustern zusammengefasst. In späteren Schritten werden jedoch auch diese Cluster mit anderen Clustern zusammengefasst, bis keine Einträge mehr zur Verfügung stehen.

### 2.4.1 UPGMA und WPGMA

Die Abkürzung UPGMA steht für *Unweighted Pairwise Grouping Method using Arithmetic Means*. Wenn die Vektoren $a$ und $b$ zu einem Cluster $c$ zusammengefasst werden, so ist die Distanz zwischen $c$ und dem Eintrag $i$ wie folgt definiert, wobei $i$ auch ein schon erzeugter Cluster sein kann:

$$d(c,i) = \frac{1}{2}(d(a,i) * d(b,i)), \tag{2.13}$$

wobei die Funktion $d(a,b)$ die Distanz zwischen den Vektoren $a$ und $b$ liefert.

Eine Erweiterung dieser Clustermethode ist WPGMA, was für *Weighted Pairwise Grouping Method using Arithmetic Means* steht. Bei der Berechnung der neuen Distanzen wird die Gleichung 2.13 verwendet, wobei die Distanz eines Vektors zu einem neuen Cluster mit der Anzahl der Elemente in diesem Cluster gewichtet wird.

### 2.4.2 *Single Linkage*

Bei der *Single Linkage* Methode wird die Distanz zwischen dem neuen Cluster $c$, welcher aus den Vektoren $a$ und $b$ gebildet wurde, und einem Eintrag $i$ definiert durch:

$$d(c,i) = min(d(a,i), d(b,i)), \tag{2.14}$$

wobei die Funktion $d(a,b)$ die Distanz zwischen den Vektoren $a$ und $b$ liefert.

### 2.4.3 *Complete Linkage*

Bei der *Complete Linkage* Methode wird die Distanz zwischen dem neuen Cluster $c$, welcher aus den Vektoren oder Clustern $a$ und $b$ gebildet wurde, und dem Eintrag $i$ definiert durch:

$$d(c,i) = max(d(a,i)\ ,\ d(b,i)), \tag{2.15}$$

wobei die Funktion $d(a,b)$ die Distanz zwischen den Vektoren $a$ und $b$ liefert.

## 2.5 Topologische Eigenschaften

Die Zentralität ist in der Graphentheorie ein Maß, um die einflussreichsten Knoten eines Netzwerkes zu identifizieren. Es existieren verschiedenste Zentralitätsmaße und je nach der Fragestellung definiert sich, welche Knoten die einflussreichsten sind. Das einfachste Zentralitätsmaß ist der Knotengrad, hier sind die Knoten einflussreich, welche die meisten Verbindungen zu anderen Knoten haben. Im Folgenden werden einige, in dieser Arbeit verwendete, Zentralitätsmaße näher erläutert.

### 2.5.1 *Closeness*-Zentralität

Die *Closeness-Zentralität* stellt die Frage, wie gut ein Knoten von allen anderen Knoten aus erreichbar ist. Dies wird für jeden Knoten durch die Reziproke der Summe der Länge aller

kürzesten Wege zwischen dem betrachteten Knoten und allen anderen Knoten angegeben:

$$C_C(x) = \frac{1}{\sum_y d(y,x)}, \tag{2.16}$$

wobei $x$ und $y$ Knoten des Graphen sind und die Funktion $d(x,y)$ die Länge des kürzesten Weges zwischen diesen beiden Knoten angibt. Je höher dieser Wert für einen Knoten ist, um so kürzer sind die Wege von anderen Knoten zu diesem. Im biologischen Kontext repräsentieren Knoten mit einer hohen *Closeness Centrality* zentrale Elemente, zum Beispiel eines Signaltransduktionsweges, da eine Regulierung dieses Elementes sich schnell auf viele andere Elemente auswirken kann.

### 2.5.2 Exzentrizität-Zentralität

Mit der Exzentrizität-Zentralität wird ebenfalls die Frage nach der guten Erreichbarkeit eines Knotens gestellt. Es wird jedoch nicht die Summe aller kürzesten Wege zwischen einem Knoten und allen anderen gebildet, sondern aus all diesen Wegen der längste ermittelt. Die Reziproke aus der Länge dieses Weges bildet das Zentralitätsmaß für einen Knoten:

$$C_E(x) = \max_{y \in V} \; d(x,y), \tag{2.17}$$

wobei $x$ und $y$ Knoten des Graphen sind und die Funktion $d(x,y)$ die Länge des kürzesten Weges zwischen diesen beiden Knoten angibt.

### 2.5.3 *Betweenness*-Zentralität

Die *Betweenness-Zentralität* betrachtet nicht die Entfernung eines Knotens zu anderen Knoten, sondern wie oft ein Knoten Teil des kürzesten Weges zwischen anderen Knotenpaaren ist. Die Formel zur Berechnung ist gegeben durch:

$$C_B(x) = \sum_{s \neq x \neq t \in V} \frac{\sigma_{st}(x)}{\sigma_{st}}, \tag{2.18}$$

wobei $\sigma_{st}$ für die Anzahl aller kürzesten Pfade zwischen den Knoten $s$ und $t$ steht und $\sigma_{st}(x)$ für den Anteil der Pfade, die durch den Knoten $x$ laufen. Eine hohe *Betweenness-Zentralität* bedeutet, dass dieser Knoten eine zentrale Rolle in der Kommunikation der Knoten spielt. Ein Knoten, der zwei Cliquen miteinander verbindet, weist einen solch hohen Wert auf, da alle kürzesten Wege zwischen allen Knotenpaaren der beiden Cliquen über diesen Verbindungsknoten laufen müssen. In einem Genregulierungsnetzwerk weist ein solch hoher Wert zum Beispiel auf einen wichtigen Regulator hin.

### 2.5.4 Eigenvektor-Zentralität

Bei der Eigenvektor-Zentralität bekommt ein Knoten einen hohen Wert zugewiesen, falls dieser mit vielen einflussreichen Knoten verbunden ist, wobei der Einfluss eines Knotens

durch seinen Knotengrad gegeben ist. Der Wert für einen Knoten wird gegeben durch:

$$C_E(x) = \frac{1}{\lambda} \sum_{t \in M(x)} C_E(t), \tag{2.19}$$

wobei $M(x)$ die Menge aller Nachbarn des Knoten $x$ bezeichnet und $\lambda$ eine Konstante ist. Durch Umstellungen in eine Vektordarstellung ist eine Darstellung als Eigenwert-Gleichung möglich:

$$\mathbf{Ax} = \lambda \mathbf{x} \tag{2.20}$$

Wobei $\mathbf{A}$ die Adjazenzmatrix des Graphen repräsentiert. Die Lösung hierfür kann mitunter mehrere Eigenwerte $\lambda$ beinhalten, für die ein Eigenvektor existiert. Daher gilt die Einschränkung, dass der Eigenvektor nur positive Einträge haben darf, was dazu führt, dass nur der größte Eigenwert für die Berechnung der Eigenvektorzentralität verwendet wird. Die Eigenvektor-Zentralität kann zum Beispiel Aufschluss darüber geben, ob und welche Bereiche eines Netzwerkes zusammenarbeiten.

## 2.6 Standards in der Systembiologie

### 2.6.1 *Systems Biology Markup Language*

Die *Systems Biology Markup Language* (SBML) (Hucka et al., 2003; Finney und Hucka, 2003) ist ein freies und offenes XML-Format (Bray et al., 1998) zum Speichern von Modellen biologischer Systeme. Inzwischen hat sich SBML zu einem Standardformat in der Systembiologie entwickelt, das von den meisten Programmen zur Modellierung biologischer Systeme unterstützt wird, und somit einen Austausch zwischen diesen ermöglicht. Zu diesem Zweck stellen die Entwickler von SBML sowohl eine Programmierschnittstelle zu vielen gängigen Programmiersprachen bereit, libSBML Bornstein et al. (2008), als auch eine spezielle Java Bibliothek, jSBML (Rodriguez et al., 2015). Das Format ermöglicht das Abbilden verschiedenster Arten von Modellen. Es gibt sowohl die Möglichkeit kinetische Parameter zu hinterlegen als auch ein rein diskretes System abzubilden. Weiter bietet das Format die Möglichkeit verschiedene Kompartimente zu definieren, um eine Lokalisierung der Elemente vorzunehmen. Die Möglichkeit zur Annotation der Elemente mittels MIRIAM (siehe Abschnitt 2.6.3) und der SBO (siehe Abschnitt 2.6.4) besteht ebenfalls.

### 2.6.2 *KEGG Markup Language*

Die KEGG Markup Language (KGML) ist ein XML-Format, welches von der KEGG Datenbank (Kanehisa und Goto, 2000) benutzt wird, um ihre Stoffwechselwege oder Signalwege zur Verfügung zu stellen (KGML, 2016). Das Format beinhaltet Informationen über alle Elemente des biologischen Systems und deren Interaktionen. Die Elemente sind nicht auf einzelne Reaktionen oder Metaboliten beschränkt, sondern können auch ganze

Enzymkomplexe, Gene oder andere Stoffwechselwege repräsentieren. Die Art der Interaktion der einzelnen Elemente kann in KGML ebenfalls differenziert werden, so stehen beispielsweise Enzym-Enzym- oder Transkriptionsfaktor-Gen-Interaktionen zur Verfügung. Diese Interaktionen können auch noch genauer definiert werden, so kann beispielsweise zwischen Dissoziation, Repression oder Phosphorylierung unterschieden werden. Weiterhin enthält das Format Informationen zum Layout des Systems. Eine genaue Beschreibung des Formats und eine Übersicht über alle unterstützten Interaktionstypen sind online zu finden (KGML-Interactions, 2016). Eine offizielle Bibliothek zum Lesen oder Schreiben von KGML Dateien steht jedoch nicht zur Verfügung.

### 2.6.3 MIRIAM

MIRIAM steht für *Minimal Information Required In the Annotation of Models* (Novere et al., 2005) und ist eine Sammlung von Richtlinien zur konsistenten Annotierung biologischer Modelle. Ziel dieser Richtlinien ist es, das Model selbst sowie jedes Element des Modells mit weiteren Informationen zu verknüpfen. Diese Verknüpfung soll Dritten eine Zuordnung der Elemente zu einem biologischen Gegenstück ermöglichen. Eine solche Zuordnung kann sowohl eine Publikation sein, welche zum Beispiel ein Enzym des Modells beschreibt, als auch ein Eintrag aus einer Datenbank, wie beispielsweise der UniProt, zu diesem Protein. Aus der entsprechenden Datenquelle (Pubmed, UniProt, usw.) und dem eindeutigen Identifikator aus jener wird ein *Uniform Resource Identifier* (URI) erstellt, der eine eindeutige Zuordnung erlaubt. Ein Beispiel für solch einen URI ist `http://identifiers.org/uniprot/P00940`, welcher zum Eintrag der Triosephosphatisomerase in der UniProt führt. MIRIAM unterstützt hierbei mehr als 550 verschiedene biologische Datenbanken, welche alle in der *MIRIAM Registry* zusammengefasst sind (MIRIAM-Registry, 2016). Zusätzlich muss die Art der Zuordnung definiert werden. Hierzu existiert eine feste Menge an Zuordnungstypen, wie zum Beispiel *isPartOf*, welcher benutzt wird, um die einzelnen Proteine eines Proteinkomplexes zu beschreiben. Eine vollständige Liste dieser vordefinierten Zuordnungstypen ist online zu finden (Qualifiers, 2016).

### 2.6.4 *Systems Biology Ontology*

Die *Systems Biology Ontology* (SBO) (Courtot et al., 2011) ist ein Vokabular von Begriffen (*SBO terms*), welche gemeinhin in der Systembiologie benutzt werden. Diese Begriffe können verwendet werden, um den Elementen eines biologischen Modells eine semantische Bedeutung zu geben. Hierzu sind die definierten Begriffe in feste Kategorien eingeteilt und besitzen alle einen eindeutigen Identifikator. Die Einteilung erfolgt in den folgenden unabhängigen Kategorien: die Rolle, die ein Element bei einer Reaktion spielen kann (z.B. „Substrat“), quantitative Parameter (z.B. „Michaelis Konstante“), Klassifizierung mathematischer Ausdrücke (z.B. „Massenwirkungszeitgesetz“), Modellierungsmethoden (z.B.

„Diskrete Modellierung“), der Art des Elements (z.B. „Macromolekül“), der Art der Interaktion (z.B. „Prozess“) und einer Kategorie zur Definition der vorhandenen Metadaten, welche dem Modell anhängig sind (z.B. anderer Annotationen). Jeder Begriff wird durch einen eindeutigen Identifikator repräsentiert. So hat zum Beispiel der Begriff „Nebenprodukt“ den eindeutigen Identifikator „SBO:0000603“ und der Identifikator „SBO:0000169“ steht für eine Inhibition.

## 2.7 Weitere Dateiformate

### 2.7.1 PNML

Die *Petri Net Markup Language* (PNML) (Weber und Kindler, 2003), ebenfalls ein XML-basiertes Dateiformat, ist im Gegensatz zu SBML auf Petrinetze spezialisiert. Dadurch ist es in der Systembiologie nicht so weit verbreitet wie SBML. Dies liegt auch mit daran, dass keine offizielle Bibliothek zum Lesen und Schreiben von PNML-Dateien zur Verfügung gestellt wird. Die letzte offizielle Version des Dateiformats erschien im Jahre 2009. Die Dokumentation dieses Formats ist nicht frei verfügbar, was eine Implementierung dieses Dateiformats ebenfalls erschwert.

### 2.7.2 PNT

Das PNT-Dateiformat (PNT, 2016) ist ein ASCII-Dateiformat, welches von der Software INA (Starke, 2003) verwendet wird und speziell für Petrinetze entwickelt wurde. Die Struktur des Petrinetzes wird durch eine Liste von Plätzen und ihren Vor- und Nach-Transitionen repräsentiert, zusätzlich mit Informationen zur Anzahl der Marken, der Kapazität und der Kantengewichte versehen. In diesem Abschnitt werden Plätze und Transitionen jedoch nur durch eindeutige Identifikatoren repräsentiert, weshalb die folgenden beiden Abschnitte die Zuordnung dieser Identifikatoren zu den Namen der Plätze bzw. den Transitionen beinhalten.

### 2.7.3 DAT

Das DAT-Dateiformat (DAT, 2016) ist ein ASCII-Dateiformat, welches von der Software MetaTool (von Kamp und Schuster, 2006) verwendet wird. Das Format wurde entwickelt, um biochemische Systeme abbilden zu können. Am Anfang des Dateiformats stehen vier Listen, welche die Namen von Reaktionen, reversiblen Reaktionen sowie die Namen interner und externer Metaboliten enthält. Im Anschluss folgt eine Liste aller vorhandenen Reaktionen.

### 2.7.4 SPPED

Das SPPED-Dateiformat ist ein XML-basiertes Dateiformat, welches von der Software Snoopy (Fieber, 2004) verwendet wird. Dieses Format wurde entwickelt, um eine große

Bandbreite verschiedener Arten von Petrinetzen zu repräsentieren. Neben der eigentlichen Struktur des PN werden in diesem Dateiformat ebenfalls Informationen über das Layout des Petrinetzes gespeichert. Es existiert keine offizielle Bibliothek zum Lesen und Erstellen von Dateien im SPPED-Format oder eine offizielle Dokumentation dieses Formats.

### 2.7.5 APNN

Die *Abstract Petri nets notation* (APNN) ist ein ASCII-Dateiformat, welches unabhängig von einer bestimmten Software entwickelt wurde (Brause et al., 1995). Im Gegensatz zu DAT oder PNT ist dieses Dateiformat zwar maschinenlesbar, für einen Menschen jedoch nicht so leicht zu lesen. Die Struktur des Dateiformats ist komplex, da nicht nur klassische P/T-Petrinetze repräsentiert werden können. Ein weiterer Unterschied zu DAT und PNT ist, dass dieses Format nicht im biologischem Kontext entwickelt wurde.

## 2.8 Serialisierung

Unter Serialisierung versteht man in der Informatik die Abbildung von Objekten in eine sequenzielle Form, um diese anschließend zum Beispiel in eine Datei oder Datenbank zu speichern. Somit kann der aktuelle Zustand der Instanz einer Klasse sowie seiner Unterobjekte gespeichert werden und später wieder hergestellt werden (Deserialisierung). Hierzu wird der Objektbaum rekursiv durchlaufen, um ein vollständiges Abbild zu erzeugen. Java bietet eine vor-implementierte Methode für diesen Prozess an, das *Interface Serializable*. Mit Hilfe dieses *Interface* wird es einfach, Objekte zu serialisieren und anschließend zu speichern. Die Objekte können in einen Bytestrom (*Java Object Serialization*) oder in ein XML-Format (*JAXB*) umgewandelt werden und anschließend in eine Datei gespeichert werden. Im Zusammenhang mit dieser Arbeit wird nur die Serialisierung in einen Bytestrom verwendet.

# Kapitel 3

# Ergebnisse und Diskussion

Im folgenden Kapitel wird die Software MonaLisa (Einloft et al., 2013) vorgestellt, welche die Modellierung von biochemischen Systemen mit Hilfe des Petrinetz-Formalismus (PN) und deren Analyse ermöglicht. MonaLisa basiert auf TInA (Thormann et al., 2009), einer Software für grundlegende Analysen von PN, wie zum Beispiel T-Invarianten. TInA wurde um eine Komponente zur Modellierung und Visualisierung von PN, zusätzliche Analysemethoden und einer Visualisierung deren Resultate erweitert. MonaLisa ist eine *Open Source* Software, der Quellcode ist unter der *Artistic Licence 2.0* (Artistic-License, 2016) im Internet verfügbar (Einloft, 2016). Die Software wurde in Java 1.7 implementiert, beinhaltet jedoch eine in C implementierte Komponente zur Berechnung der Invarianten. Neben der eigentlichen Modellierung und Analyse von PN ist die Grundidee von MonaLisa, dass alle Resultate der verschiedenen Analysemethoden für ein besseres Verständnis auf dem PN visuell repräsentiert werden können. Vor allem bei komplexeren Modellen und steigender Anzahl an Resultaten wird so eine Auswertung und Interpretation dieser unterstützt. So soll dem Modellierer geholfen werden, die Resultate im biologischen Kontext zu sehen, zu verstehen und zu bewerten. Um MonaLisa einfach um neue Analysemethoden erweitern zu können, wurde eine entsprechende modulare Klassenstruktur geschaffen. MonaLisa unterstützt viele gängige externe Dateiformate, wodurch der Austausch von Modellen mit anderen Programmen ermöglicht wird.

Im Folgenden werden zunächst die wichtigsten Klassen von MonaLisa und deren Implementierung beschrieben. Im Anschluss daran erfolgt eine Übersicht über die verschiedenen Komponenten von MonaLisa und den verfügbaren Analysemethoden.

In diesem Kapitel findet auch eine Diskussion der Ergebnisse statt. Im Rahmen dieser Diskussion wird ebenfalls ein Ausblick auf weitere mögliche Entwicklungen gegeben, um diese nicht in einem eigenen Abschnitt aus dem Zusammenhang zu reißen.

## 3.1 Implementierung essentieller Klassen

### 3.1.1 Implementierung von Klassen zur Repräsentation eines Petrinetzes

Damit mit MonaLisa ein PN analysiert werden kann, wird zunächst eine Klassenstruktur benötigt, welche ein solches abbilden kann und einen Zugriff auf dessen Struktur ermöglicht. Dafür wurde das Klassenpaket *monalisa.data.pn* implementiert, welches in Abbildung 3.1 als UML-Klassendiagramm gezeigt wird. Die Klassen werden aus Platzgründen nicht vollständig dargestellt. Grundlage aller Klassen dieses Paketes ist die abstrakte Klasse *AbstractPetriNetEntitiy*, deren Aufgabe das Bereitstellen einer *PropertyList* ist. Dies ist eine Klasse, die es mit Hilfe einer Hashtabelle *HashMap<String,Object>* ermöglicht, beliebige Informationen, unabhängig vom Datentyp, zu speichern. So können für ein Element

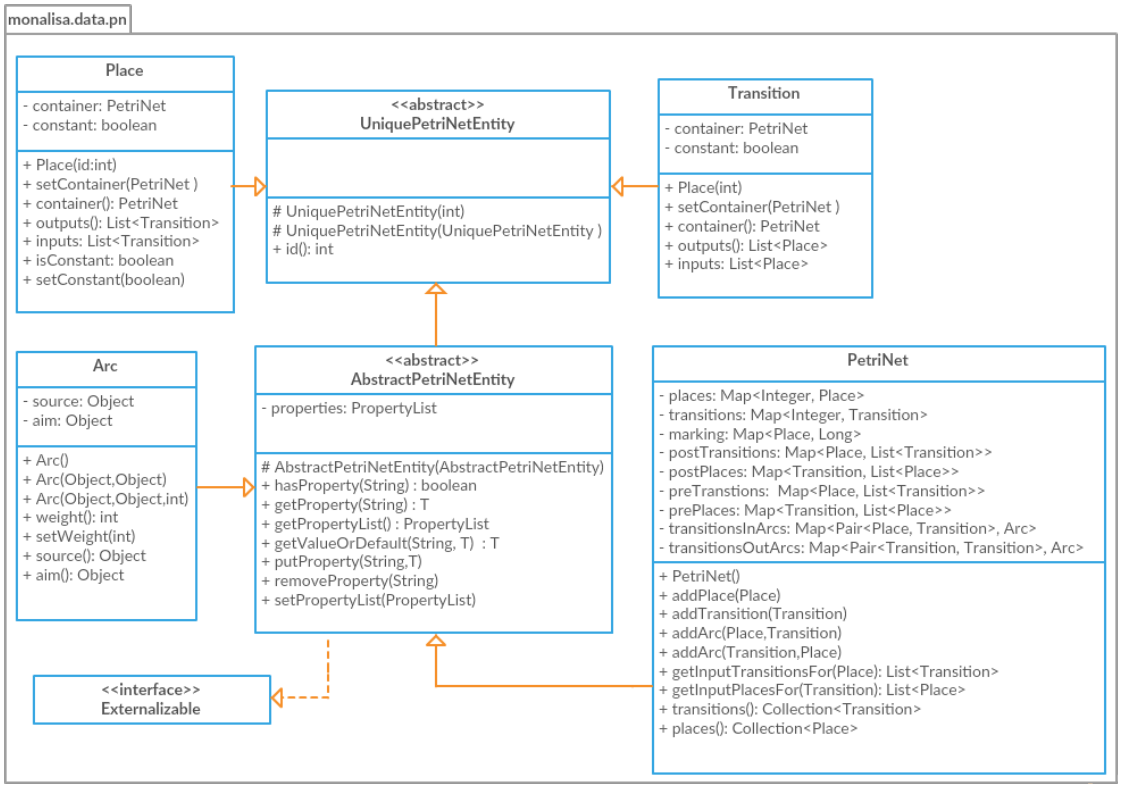


**Abbildung 3.1:** Die Abbildung zeigt in Auszügen das UML-Klassendiagramm der Klassen, welche der Repräsentation eines PN dienen. Grundlage aller Klassen bildet die abstrakte Klasse *AbstractPedtriNetEntity*, welche eine *PropertyList* bereit stellt, in welcher beliebige Objekte gespeichert werden können. Vor den Klassen für die Plätze (*Place*) und Transitionen (*Transition*) ist die abstrakte Klasse *UniquePetriNetEntity* geschaltet. Diese führt einen eindeutigen Identifikator für Plätze und Transitionen ein. Informationen zu Vorplätzen oder Nachtransitionen werden nicht direkt in diesen Klassen gespeichert, sondern zentral von der *PetriNet* Klasse verwaltet. Hierzu existieren in dieser Klasse interne Listen zum Speichern dieser Informationen und Funktionen zur Abfrage dieser. Die Klasse *Arc* dient der Speicherung der Kanten, für die jeweils der Start- und Endpunkt und das Gewicht der Kante gespeichert wird.

des PN bei Bedarf beliebige Informationen gespeichert werden, ohne dass der entsprechenden Klasse ein neues Attribut hinzugefügt werden muss. Die *PropertyList* sorgt so für eine Flexibilität der Klassen, welche für das spätere Erweitern von MonaLisa um neue Analysemethoden nötig ist. Den Klassen zur Repräsentation der Plätze und Transitionen ist eine weitere abstrakte Klasse vorgeschaltet, die Klasse *UniquePetriNetEntity*. Die Aufgabe dieser Klasse ist es, für Plätze und Transitionen einen eindeutigen Identifikator zur Verfügung zu stellen. Beide Klassen, *Transition* und *Place*, stellen Funktionen zur Verfügung, wie zum Beispiel für das Abfragen aller Vor- oder Nachplätze. Die Informationen hierfür werden jedoch nicht in den Klassen, sondern zentral in der Klasse *PetriNet* gespeichert und von den *Place*- oder *Transition*-Objekten jeweils dort abgefragt. Zu diesem Zweck speichern diese Klassen im Attribut *container* die Instanz des *PetriNet*-Objektes, zu dem sie gehören. Die Idee hinter diesem Vorgehen ist, dass die Informationen zu Nachbarschaften zentral gespeichert und verwaltet werden können und somit nicht redundant in jedem Platz oder jeder Transition hinterlegt sind. Gleichzeitig ist dieses Konzept auch notwendig, da die interne PN-Repräsentation durch Serialisierung gespeichert wird. Würde eine Instanz der Klasse *Place* Referenzen zu seinen Vortransitionen besitzen, entstünde eine Schleife im Klassenbaum, da es in den Transitionen wiederum Referenzen auf die Instanz der Klasse *Place* gibt (als Nachplatz dieser Transitionen). Solche Schleifen führen bei einer Serialisierung dieser Objekte zu Endlosschleifen und somit zu einem Fehler. Die Klasse *Arc* dient zur Repräsentation der Kanten des PN. Diese Klasse speichert neben Start- und Endpunkt auch das Gewicht der Kante.

Diese Klassenstruktur dient der Repräsentation des Quintupels eines PN und dient als Grundlage für einen Großteil der Analysemethoden von MonaLisa. Aus diesem Grund ist sie ein wichtiges Grundelement von MonaLisaund Fehler in der Klassenstruktur führen zu falschen Resultaten. Um diese zu vermeiden, wurde es notwendig, die Klassen mit Testklassen, sogenannten *JUnit*-Tests, auf Fehler zu überprüfen. Diese Testklassen wurden in Zusammenarbeit mit Sonja Scharf entwickelt (Scharf, 2015) und können nach Code-Änderungen an PN-Klassen automatisch ausgeführt werden, um deren korrekte Funktionalität zu überprüfen.

Neben P/T-Systemen existieren viele Erweiterungen des PN-Formalismus, wie zum Beispiel gefärbte PN, zeitbehaftete PN oder hybride PN. Die Modellierung solcher PN-Erweiterungen ist mit MonaLisa zur Zeit noch nicht möglich. Im Gegensatz zu Programmen wie beispielsweise Snoopy (Fieber, 2004) war die anfängliche Motivation hinter MonaLisa nicht die Unterstützung möglichst vieler PN-Varianten, sondern das Bereitstellen möglichst vieler Analysemethoden, speziell für biologische Netzwerke. Neben der Implementierung weiterer Analysemethoden im Rahmen einer künftigen Weiterentwicklung von MonaLisa könnte die Software auch um solche PN-Varianten ergänzt werden. Für diesen Schritt wird keine vollständig neue Klassenstruktur benötigt, die vorhandenen Klassen können als Ausgangspunkt für neue Klassen verwendet werden. Neue PN-Varianten können ebenfalls auf den vorhandenen Klassen aufbauen und diese erweitern.

### 3.1.2 Implementierung der Interfaces *Tool*, *Configuration* und *Result*

Für eine einfache Implementierung und Handhabung der Algorithmen zur Analyse von PN und deren Resultate wurden die Interfaces *Tool*, *Configuration* und *Result* implementiert. Das UML-Klassendiagramm dieser wird in Auszügen in Abbildung 3.2 gezeigt. Die Grundlage der meisten implementierten Analysemethode ist das Interface *Tool*. In diesem werden Funktionen vordefiniert, welche für das Starten der Analysemethode, deren Durchführung und der Abfrage des Fortschritts dieser verantwortlich sind. Daneben werden auch Funktionen zum Abfragen der Voraussetzungen einer Analysemethode oder eine Funktion, welche nach Beendigung der Analyse aufgerufen wird, vorgegeben. Da die meisten dieser Funktionen eine identische Implementierung in den einzelnen Analysemethoden hätten, wird diesen die abstrakte Klasse *AbstractTool* vorgeschaltet. In dieser abstrakten Klasse werden Funktionen vorimplementiert, die zum Abfragen des Fortschritts der Analyse dienen. Zur Implementierung einer Analysemethode ist es jetzt nur noch notwendig, zu definieren, welche Voraussetzungen diese benötigt, wie zum Beispiel das Vorhandensein von T-Invarianten oder die Auswahl bestimmter Parameter und die Implementierung der

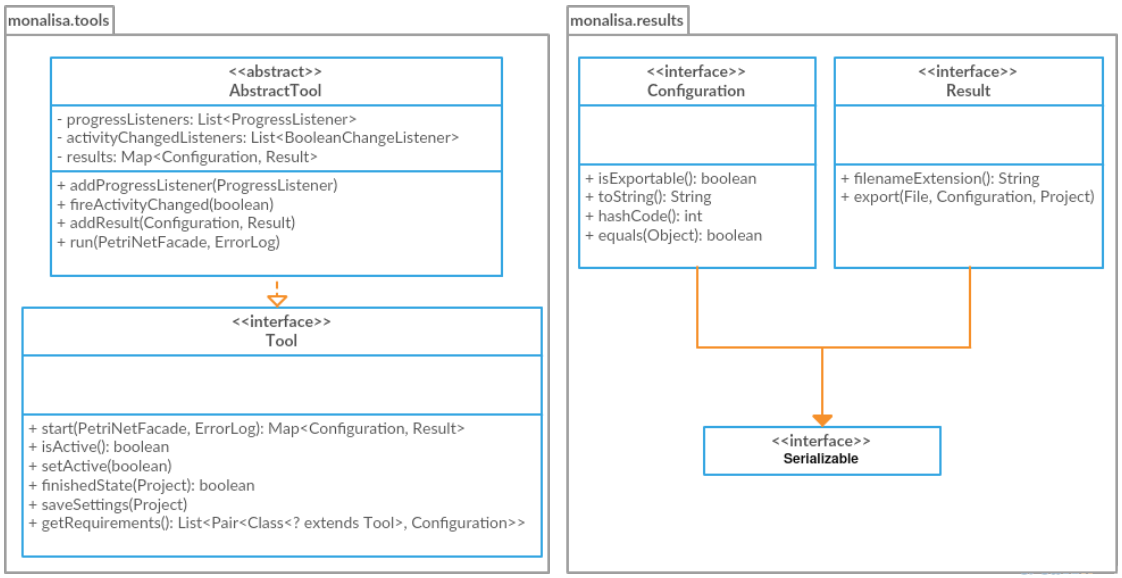


**Abbildung 3.2:** Die Abbildung zeigt in Auszügen die UML-Klassendiagramme der Interfaces *Tool*, *Configuration* und *Result*. Das Interface *Tool* definiert einige grundlegende Funktionen, welche eine Analysemethode zu implementieren hat. Dazu gehören unter anderem Funktionen zum Starten der Analysemethode und zum Abfragen seiner Voraussetzungen. Eine Analysemethode implementiert jedoch nicht dieses Interface, sondern erweitert die abstrakte Klasse *AbstractTool*. In dieser sind schon einige Funktionen vorimplementiert, die in jeder Analysemethode identisch wären, wie zum Beispiel das Hinzufügen eines Resultats zum Projekt. Das Interface *Configuration* dient als Grundlage für Klassen, in denen die Einstellungen zu einer Analysemethode verwaltet werden können. Es definiert jedoch nur sehr wenige Funktionen, zum Beispiel zum Abfragen, ob die Resultate der Analysemethode exportierbar sind. Diese Resultate werden in Klassen gespeichert, welche das Interface *Result* implementieren. Die interne Repräsentation dieser ist dabei völlig frei, vorgegeben sind zwei Funktionen, die dem Export dieser Resultate dienen. Die Interfaces *Result* und *Configuration* implementieren jeweils das Interface *Serializable*.

Algorithmen zur Analyse selbst. Zusätzlich wird den Analysemethoden die Möglichkeiten einer kleinen grafischen Oberfläche gegeben.

Einige Analysemethoden verfügen über Parameter, zum Beispiel das zu verwendende Distanzmaß zur Berechnung der T-Cluster. Das Interface *Configuration* dient dazu, solche Parameter zu definieren und für das Durchführen der Analyse zur Verfügung zu stellen. Da nicht vorhergesagt werden kann, welche Parameter eine Analysemethode benötigt, ist dieses Interface sehr schlicht gehalten. Es gibt unter anderem vor, dass die *equals*-Funktion implementiert werden muss sowie die dazugehörige Funktion *hashCode*, um verschiedene Parameterkombinationen eindeutig unterscheiden zu können. Welche Parameter gespeichert werden und wie der Zugriff auf diese erfolgt, ist der Implementierung dieses Interfaces völlig frei gelassen.

Die Resultate einer Analysemethode werden in Klassen gespeichert, die das Interface *Result* implementieren. Das Speichern der Resultate an sich ist dabei jeder Implementierung frei überlassen, da auch hier nicht vorhergesagt werden kann, welche Form diese haben werden. Lediglich zwei Funktion zum Export der Resultate in eine Datei werden vom Interface definiert.

Durch die in diesem Abschnitt beschriebenen und in MonaLisa implementierten Strukturen können schnell und einfach neue Analysemethoden zu MonaLisa hinzugefügt werden. Sollte, wie beschrieben, MonaLisa um neue Varianten von PN erweitert werden, muss das Interface *Tool* ebenfalls angepasst werden. Für eine Analysemethode müsste hier zusätzlich vermerkt sein, für welche PN-Implementation diese ausgelegt ist. Auch hier würde das Schaffen einer komplett neuen Klassenstruktur entfallen.

### 3.1.3 Implementierung einer Visualisierung von Petrinetzen

Die Basis von MonaLisa ist die Software TInA, die jedoch keine Möglichkeit zur Modellierung und Visualisierung von PN zur Verfügung stellt. Aus diesem Grund wurde für MonaLisa eine neue Komponente entwickelt, die genau dies ermöglicht. Diese Komponente wird *NetViewer* genannt und benötigt eine gesonderte interne Repräsentation des PN.

Die Visualisierung des PN im *NetViewer* erfolgt mit Hilfe der Java-Bibliothek *Java Universal Network/Graph Framework*, JUNG (JUNG, 2016). Diese Bibliothek dient zur Visualisierung von Graphen und stellt alle dafür nötigen Klassen und Funktionen zur Verfügung. Durch starke Modularisierung ist es möglich, die Visualisierung der Graphen sehr einfach tief-greifend zu beeinflussen. Die Bibliothek stellt verschiedene Graph-Klassen bereit, wie zum Beispiel *SparseGraph, HyperGraph, Forest*, welche die Topologie der Graphen enthalten. Die für die Visualisierung nötigen Informationen, wie zum Beispiel die Koordinaten eines Knotens, werden von einer *Layout*-Klasse verwaltet. Die Instanz der Klasse *Graph* und der dazugehörigen Klasse *Layout* wird an einen *VisualizationViewer* übergeben, welcher nun für die Visualisierung des Graphen auf dem Bildschirm und für die Interaktion mit dem Benutzer verantwortlich ist. Der *VisualizationViewer* wiederum

nutzt *Renderer*-Klassen, um die verschiedenen Aspekte der Visualisierung, wie zum Beispiel Farbe oder Form eines Knotens, festzulegen. Durch Überschreiben dieser Klassen ist es dem Programmierer sehr leicht möglich, diese Eigenschaften zu beeinflussen. Diese können global für alle Knoten oder aber für jeden Knoten einzeln geändert werden. Die Klassen unterliegen hierbei dem Prinzip der generischen Programmierung. Ein bekanntest Beispiel für dieses Konzept ist die Java Klasse *List*, in welcher nur Elemente eines Datentyps gespeichert werden. Da aber nicht für jeden Datentyp eine eigene Implementierung der Klasse *List* vorliegen kann, wird der Datentyp der Liste erst beim Initialisieren dieser gesetzt, zum Beispiel *List<Integer>*. Genauso verhält es sich bei der Klasse *Graph* der JUNG-Bibliothek. Der Datentyp der Klasse für die Knoten und Kanten ist erst einmal nicht bekannt, denn der Entwickler kann diese frei wählen. Im Falle des *NetViewer* werden hierzu jedoch nicht die vorhandenen Klassen der PN-Repräsentation verwendet. Zum Einen liegen hier zwei Klassen für die Knoten des PN vor, *Place* und *Transition*, von JUNG wird aber nur ein Klasse für Knoten unterstützt. Zum Zweiten werden für die Visualisierung andere Anforderungen gestellt, da viel mehr Informationen in den Knoten gespeichert werden müssen, wie zum Beispiel dessen Farbe oder Form. Und zum Dritten sollte das Anlegen logischer Plätze möglich sein, also eine Kopie eines Platzes, welcher nur mit bestimmten Knoten verbunden ist. Ein solcher Platz existiert in der Visualisierung mehrmals, im darunterliegenden PN aber nur einmal. Diese drei Aspekte führten dazu, dass von einer Erweiterung der vorhandenen Klassen abgesehen wurde, zumal es sich so vermeiden ließ, die vorhandenen Klassen, welche auf das PN zugreifen, anzupassen. Stattdessen wurden die beiden Klassen *NetViewerNode* und *NetViewerEdge* implementiert, deren UML Diagramm in Auszügen in Abbildung 3.3 zu sehen sind.

Zur Unterscheidung zwischen Transitionen und Plätzen besitzt die Klasse *NetViewerNode* das Attribut *nodeType*, welches diese Zuordnung zulässt. Weiter werden in dieser Klasse alle Informationen zur Visualisierung des Knotens gespeichert. Dazu zählen unter anderem die Füllfarbe, die Farbe der Umrandung und die Form. Ein weiteres Attribut gibt Auskunft, ob es sich um einen logischen Platz handelt. Ist dies der Fall, existiert ein *Master Node*, welcher den ursprünglichen Platz repräsentiert. Dieser kennt alle seine logischen Plätze sowie jeder logischen Plätze seinen *Master Node* kennt. Denn wird zum Beispiel die Farbe eines logischen Platzes geändert, wird diese für alle anderen Plätze gleichen Namens ebenfalls übernommen.

Die Klasse *NetViewerEdge* speichert neben dem Kantengewicht ebenfalls Informationen zur Farbe einer Kante. Wird ein Knick in eine Kante eingebaut, so geschieht dies durch das Platzieren eines unsichtbaren Knotens an dieser Stelle und das Aufteilen der Kante in zwei neue Kanten. Die ursprüngliche, direkte Kante, wird jedoch nicht gelöscht, sondern lediglich unsichtbar gemacht. Das Löschen dieser Kante würde dazu führen, dass über die *Graph*-Klasse nicht mehr die Nachbarschaft zweier echter Knoten abgefragt werden könnte. Wie bei den logischen Plätzen, kennt jede so erzeugte Zwischenkante seine *Master Edge*. Als *Graph*-Klasse wird in MonaLisa die Klasse *SparseGraph* verwendet.

Die hier beschriebenen Klassenstrukturen ermöglichen die Visualisierung eines PN und

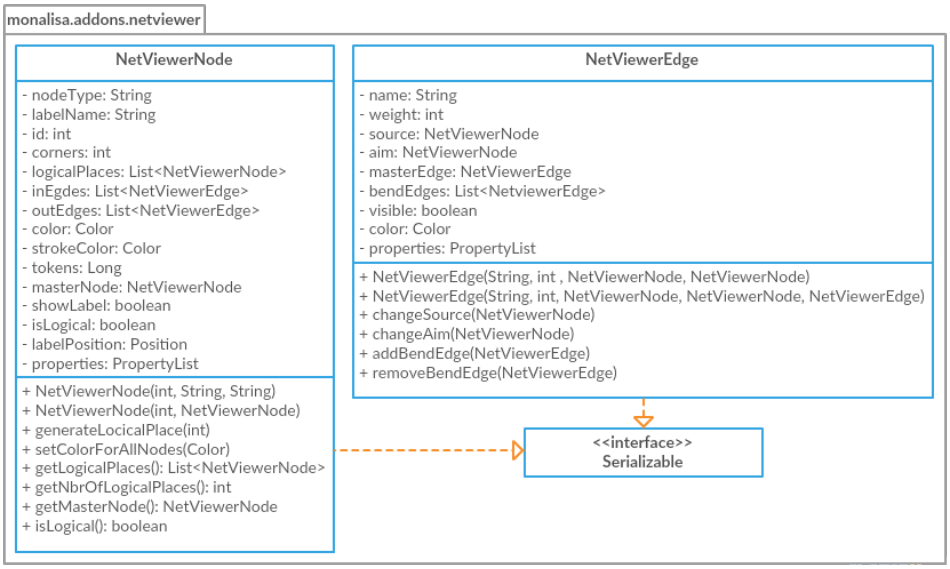


**Abbildung 3.3:** Die Abbildung zeigt in Auszügen die UML-Klassendiagramme der beiden Klassen *NetviewerNode* und *NetViewerEdge*. Diese beiden Klassen werden im *NetViewer* verwendet, um die grafische Repräsentation eines PN zu realisieren. In der Klasse *NetViewerNode* dienen die meisten Attribute dazu, die grafischen Eigenschaften eines Platzes oder einer Transition zu speichern, so unter anderem die Füllfarbe, die Farbe der Umrandung, die Form oder die Position der Beschriftung. Daneben gibt es weitere Attribute und Funktionen, die dem Verwalten der logischen Plätze dienen. In der Klasse *NetViewerEdge* wird die Farbe und das Gewicht einer Kante gespeichert sowie Informationen zu eingebauten Knicken in dieser. Beide Klassen implementieren das Interface *Serializable*.

umfangreiche Möglichkeiten zum Anpassen dieser. Die Darstellung der einzelnen Elemente kann ebenso beeinflusst werden wie das Layout durch Hinzufügen logischer Plätze und Knicke in Kanten. Logische Plätze ermöglichen eine Verbesserung des Layouts, da durch diese die Anzahl kreuzender Kanten reduziert werden kann. Ein weiteres Element, welches bei der besseren Organisation des Layouts helfen kann, sind hierarchische Transitionen. Eine solche stellt in der Visualisierung des PN eine Transition dar, hinter der jedoch eine größere Struktur an Plätzen und Transitionen verborgen ist, die auf diese Weise versteckt wird. Sie agiert also als *Blackbox*. So können größere oder kleine Substrukturen eines Modells zwar modelliert werden, ihre grafische Darstellung jedoch reduziert werden, um ein übersichtlicheres Layout zu erhalten. Das Erweitern von MonaLisa um hierarchische Transitionen wäre ein nächster Schritt für die Weiterentwicklung der Software. Hierzu ist jedoch ein größerer Eingriff in die Funktionalitäten der Klassen für die Visualisierung des PN nötig. So muss zum Beispiel die Zuordnung einzelner Elemente des PN zu der Substruktur einer hierarchischen Transition ermöglicht werden. Weiter muss eine Möglichkeit geschaffen werden, wie die Elemente dieser Substrukturen modelliert und editiert werden oder bedacht werden können.

Sollte, wie beschrieben, MonaLisa um weitere PN-Varianten erweitert werden, so wird ebenfalls eine Anpassung der Visualisierung der PN nötig. Wie bei den Klassen zur

Repräsentation eines PN würde auch hier keine komplett neue Klassenstruktur notwendig, und die vorhandenen Klassen können als Ausgangspunkt der neuen Klassen verwendet werden. Dank der generischen Klassen von JUNG wäre die Implementation neuer Klassen für Transitionen, Plätze und Kanten, um zum Beispiel inhibitorische Kanten oder zeitabhängige Transitionen zu repräsentieren, sehr einfach. Für die Visualisierung dieser neuen Elemente müssten neue *Renderer* Klassen implementiert werden.

### 3.1.4 Implementierung einer Synchronisation zwischen dem PN und der Visualisierung

Die in den beiden Abschnitten 3.1.1 und 3.1.3 beschrieben Ebenen der Repräsentation des PN machen es nötigt, beide Ebenen synchron zu halten, damit beide zu jedem Zeitpunkt dasselbe PN abbilden. Es handelt sich hierbei um eine einseitige Synchronisation, da der Benutzer nur mit der Visualisierung des PN interagiert und Änderungen nur an diesem stattfinden und anschließend an die eigentliche PN-Klassen weitergegeben werden müssen. Diese Aufgabe übernimmt die Klasse *Synchronizer*, die entsprechende UML Darstellung ist in Auszügen in Abbildung 3.4 dargestellt. Kern dieser Klasse sind die aktuellen Referenzen des *PetriNet*- und des *Graph*-Objekts. Die Klasse stellt Funktionen zum Hinzufügen oder Löschen neuer Knoten sowie eine Vielzahl weiterer Funktionen zur Verfügung. Die Funktionen führen die entsprechenden Operationen sowohl im *PetriNet*-Objekt als auch im *Graph*-Objekt durch. Durch das Verwenden dieser beiden Klassen ist es zudem nötig, eine eindeutige Zuordnung eines *Place*-Objekts oder eines *Transition*-Objekts auf das entsprechende *NetViewerNode*-Objekt und umgekehrt bereitzustellen. Gleiches gilt für die eindeutige Zuordnung eines *Arc*-Objekts zu dem entsprechenden *NetViewerEdge*-Objekt. Um dies zu ermöglichen, existieren mehrere interne Listen und Hashtabellen. Wird ein vorhandenes Modell über ein externes Dateiformat in MonaLisa importiert, wird dieses mit der Funktion *translatePNtoGraph()* von einem *PetriNet*-Objekt in ein *Graph*-Objekt übersetzt. Stellt das externe Dateiformat Informationen zum Layout bereit, werden diese verwendet. Ansonsten wird automatisiert ein Layout mit Hilfe der JUNG-Bibliothek erstellt.

Die Trennung der Klassen zur Repräsentation des PN und der Visualisierung dieses entspringt den unterschiedlichen beschriebenen Anforderungen an diese Klassen. Die Visualisierung eines PN benötigt mehr Informationen und Eigenschaften als die einfache Repräsentation der Topologie eines PN. Zudem machen Funktionen wie logische Plätze oder Knicke in Kanten das Hinzufügen zusätzlicher Knoten in der Visualisierung nötig, weshalb die Visualisierung eine andere Topologie als das eigentliche PN besitzen kann.

Eine Folge aus dieser Trennung ist die Notwendigkeit der Synchronisation beider Strukturen. Kommt es hier zu Fehlern, passen die Visualisierung und das PN nicht mehr zueinander, was zu fehlerhaften Resultaten und Interpretationen führt. Dadurch ist die Synchronisation beider Klassenstrukturen einer der kritischen Punkte in MonaLisa. Erleichtert wird diese dadurch, dass eine Veränderung an der Topologie des PN immer nur in der

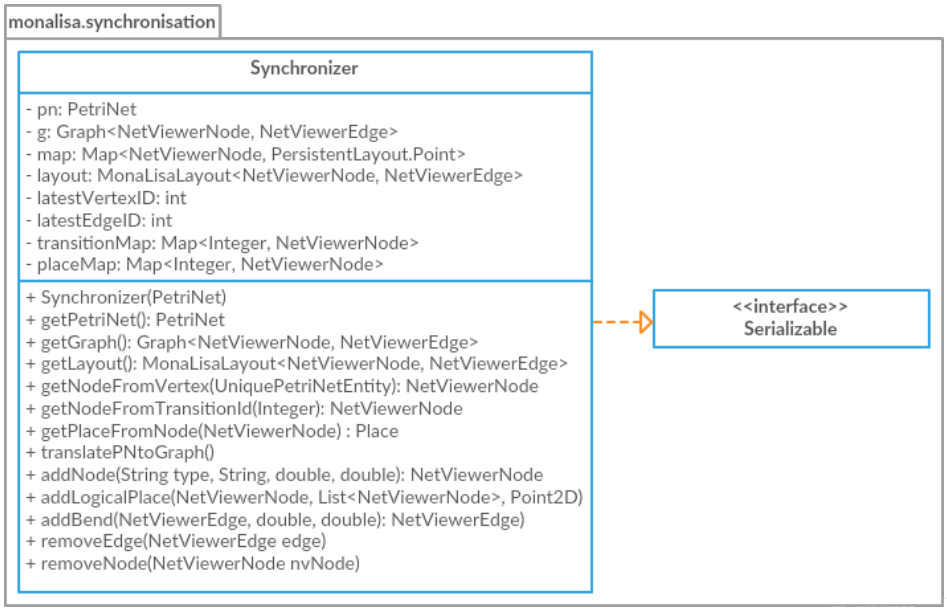


**Abbildung 3.4:** Die Abbildung zeigt in Auszügen das UML-Klassendiagramm der Klasse *Synchronizer*. Diese Klasse ist die Schnittstelle zwischen den PN-Klassen, dem Attribut *pn* und der grafischen Repräsentation dieser, gegeben durch die beiden Attribute *g* und *layout*. Neben diesen beiden Attributen existieren interne Hashtabellen und entsprechende Funktionen, um eine eindeutige Zuordnung eines *Place*- oder *Transition*-Objekts auf das entsprechende *NetViewerNode*-Objekt zu ermöglichen und umgekehrt. Die Klasse implementiert das Interface *Serializable*.

Visualisierung stattfindet. Die Klassen zur Repräsentation eines PN wurden mit *JUnit*-Tests abgedeckt. Im Rahmen der Bachelorarbeit von Sonja Scharf (Scharf, 2015) wurde ebenfalls begonnen, die Klasse *Synchronizer* mit *JUnit*-Tests abzudecken. Die zentrale Problemstellung hier ist, dass für einen Test der aktuellen Werte der Klasse zwei Graphen miteinander verglichen werden müssen. Dafür muss das Problem des Isomorphismus zweier Graphen gelöst werden. Der Fokus der Bachelorarbeit lag jedoch auf der Abdeckung der PN-Klassen, weshalb dieses Problem nicht gelöst wurde. Für eine Weiterentwicklung von MonaLisa wäre die vollständige Abdeckung der *Synchronizer*-Klasse mit *JUnit*-Tests von großem Interesse, da dann ein weiteres zentrales Element von MonaLisa automatisch auf Fehler überprüft werden kann.

Für die Einführung neuer PN-Varianten wäre auch die Implementierung weiterer *Synchronoizer*-Klassen notwendig.

### 3.1.5 Implementierung der abstrakten Klasse *AddonPanel*

Die Analysemethoden, welche auf dem Interfaces *Tool* basieren, haben keine Möglichkeit, direkt mit der Visualisierung des PN im *NetViewer* zu interagieren oder eine aufwändigere grafische Oberfläche bereitzustellen. Aus diesem Grund bietet der *NetViewer* ebenfalls die Möglichkeit, weitere Analysemethoden zu integrieren. Eine solche Analysemethode muss die abstrakte Klasse *AddonPanel* erweitern, welche in Abbildung 3.5 in Auszügen als

UML-Klassendiagramm gezeigt wird. Ihre Aufgabe ist es, die grafische Oberfläche der Analysemethode in die des *NetViewer* zu integrieren, die Interaktion mit der Visualisierung zu ermöglichen und den Zugriff auf die *PetriNet*-Klasse zu ermöglichen. Letzteres geschieht auch hier über die zwischengeschaltete Klasse *PetriNetFacade*. Eine weitere Aufgabe der Klasse *AddonPanel* ist es, den Analysemethoden zu ermöglichen, ihre Resultate oder Einstellungen in der Projektdatei von MonaLisa zu speichern. Zu diesem Zweck implementiert die abstrakte Klasse *AddonPanel* das Interface *AddonStorageManagment*. Eine genaue Beschreibung dieses Interfaces folgt im Abschnitt 3.1.7. Ebenfalls implementiert wird das Interface *NetChangedListener*. Über dieses Interface kann der *NetViewer* den Erweiterungen mitteilen, dass sich die Topologie des PN geändert hat und somit vorhandene Resultate nicht mehr gültig sind.

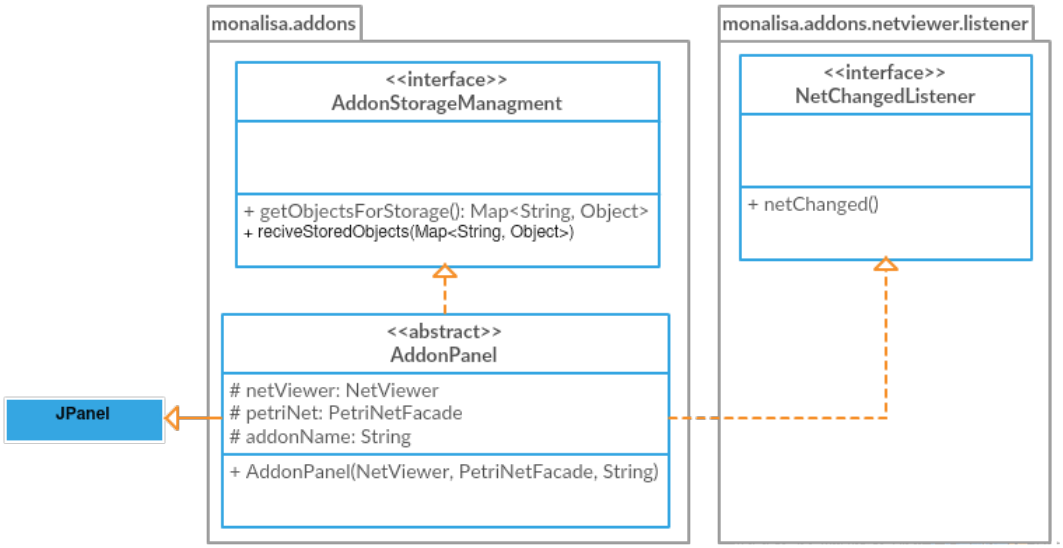


**Abbildung 3.5:** Die Abbildung zeigt in Auszügen das UML-Klassendiagramm der abstrakten Klasse *AddonPanel* und der von ihr implementierten Interfaces. Die abstrakte Klasse *AddonPanel* dient als Grundlage von Analysemethoden, welche den *NetViewer* für die Anzeige ihrer grafischen Oberfläche nutzen. Daher erweitert sie die Java-Klasse *JPanel*. Diesen Analysemethoden stellt die Klasse ein aktuelles Objekt der Klasse *PetriNetFacade* und des *NetViewer* selbst zur Verfügung. Damit die Erweiterungen auf Änderungen im PN reagieren können, implementiert diese Klasse das Interface *NetChangedListener*. Das Interface *AddonStorageManagment* wird implementiert, damit die Analysemethode Einstellungen und Resultate in der Projekt-Datei speichern kann.

Mit der Klasse *AddonPanel* und ihren implementierten Interfaces wird eine weitere Möglichkeit zum Erweitern von MonaLisa um neue Analysemethoden bereitgestellt. Das Interface *AddonStorageManagment* ermöglicht für diese Analysemethoden auch das Speichern von Resultaten und Einstellungen in der zentralen Projektdatei von MonaLisa. Die Analysemethoden können direkt mit der Visualisierung des PN interagieren und ermöglichen so weitergehende Untersuchungen des PN. Anwendungen der Klasse *AddonPanel* werden im weiteren Text aufgezeigt.

Mit dieser Klasse können Analysemethoden zur Zeit nur an einer Stelle in die grafische Oberfläche eingebunden werden, nämlich in der *ToolBar* des *NetViewer*. Für eine Weiterentwicklung von MonaLisa könnten zur Einbindung der grafischen Oberfläche von Analysemethoden weitere Möglichkeiten geschaffen werden. So wäre zum Beispiel eine Einbindung in die Menüleiste des *NetViewer* oder in die Kontextmenüs des *NetViewer* denkbar. Hierzu wäre die Implementation neuer Klassen analog zur Klasse *AddonPanel* nötig.

### 3.1.6 Implementierung der Klasse *PetriNetFacade*

Die *PetriNet*-Klasse ist eine zentrale Klasse zur Analyse der mit MonaLisa erstellten PN. Änderungen am PN werden nur im *NetViewer* vorgenommen und dann weitergegeben. Würden Analysemethoden vollen Zugriff auf die *PetriNet*-Klasse erhalten, könnten sie die *Synchronizer*-Klasse umgehen und Änderungen am PN vornehmen, ohne dass dies in der Visualisierung erkenntlich wäre. Um dies zu verhindern, wurde die Fassaden-Klasse *PetriNetFacade* implementiert. Diese stellt nur diejenigen Funktionen der *PetriNet*-Klasse bereit, die das Abfragen von Informationen ermöglichen, aber keine Manipulation dieser erlauben. Die Klasse *PetriNetFacade* verwendet die gleichen Funktionsnamen und -signaturen wie die *PetriNet*-Klasse und leitet deren Aufrufe an diese weiter.

Durch die Klasse *PetriNetFacade* wird den einzelnen Analysemethoden die Möglichkeit genommen, das PN zu manipulieren. Ist es für eine Methode jedoch notwendig, Änderungen am PN vorzunehmen, so können diese sich eine Kopie des PN anfertigen und die Änderungen an dieser vornehmen. Ein solches Vorgehen ist zum Beispiel für die Durchführung der Knock-out-Analysen notwendig. Eine mögliche Einführung weiterer PN-Varianten in MonaLisamacht eine Anpassung der Klasse *PetriNetFacade* nötig. Für eine neue PN-Variante wird eine neue Basis-Klasse nötig, für welche dann jeweils eine neue Fassaden-Klasse nötig wäre.

### 3.1.7 Implementierung der Klasse *Project*

Die mit MonaLisa erstellten PN sollen für eine spätere Verwendung gespeichert werden können. Neben der reinen Struktur des PN sollen auch die grafische Repräsentation und bereits berechnete Resultate gespeichert werden können. All diese Informationen werden zentral in der Klasse *Project* zusammengefasst, die anschließend über Serialisierung gespeichert und wieder eingelesen werden kann. Das UML-Klassendiagramm der Klasse *Project* wird in Auszügen in Abbildung 3.6 dargestellt.

Zur Abspeicherung der vorhandenen Resultate wird eine Hashtabelle *Map<Class<? extends Tool >, Map<Configuration, Result>>* verwendet. Alle Informationen über das PN und dessen grafische Darstellung werden über die aktuelle Referenz der *Synchronzier*-Klasse bereitgestellt. Die Analysemethoden, welche auf der abstrakten Klasse *AddonPanel* aufbauen, können ebenfalls Daten in der Klasse *Project* speichern. Hierzu existiert

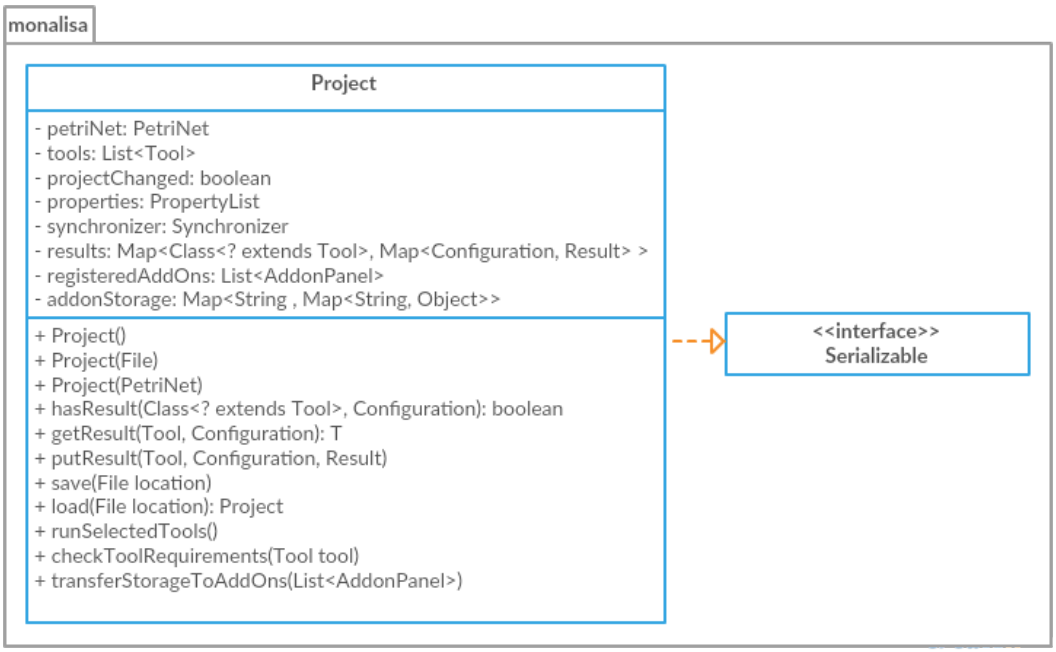


**Abbildung 3.6:** Die Abbildung zeigt in Auszügen das UML-Klassendiagramm der Klasse *Project*. In dieser Klasse laufen alle Informationen zusammen, die zum Speichern und Laden eines MonaLisa-Projektes benötigt werden. Das aktuelle PN und dessen grafische Repräsentation wird durch das Attribut *synchronizer* bereitgestellt. Die Resultate der verschiedenen Analysemethoden werden über die Attribute *results* und *addonStorage* gespeichert. Zudem stellt die Klasse Funktionen bereit, um Projekte aus dem Import externer Dateiformate zu erstellen. Auch das Starten und Verwalten der Analysemethoden, welche auf dem Interface *Tool* basieren, findet in dieser Klasse statt. Die Klasse implementiert das Interface *Serializable*.

eine Hashtabelle *Map<String , Map<String, Object>>*. *String* steht für den Namen einer Analysemethode, für die in einer *Map<String, Object>>* beliebige Objekte gespeichert werden können, die wiederum mit einem Namen versehen sind. Beim Speichern des aktuellen Objektes der *Project*-Klasse müssen bei diesen Analysemethoden die zu speichernden Daten abgerufen werden und beim Deserialisieren wieder an diese weitergegeben werden. Zu diesem Zweck implementiert die abstrakte Klasse *AddonPanel* das Interface *AddonStorageManagment*. Wird ein Projekt gespeichert, so wird die Funktion *getObjectsForStorage()* in jeder Erweiterung aufgerufen. In dieser Funktion muss jede Erweiterung die Daten bereitstellen, welche gespeichert werden sollen. Im umgekehrten Fall wird die Funktion *reciveStoredObjects()* benutzt, um den Erweiterungen die gespeicherten Daten zu übergeben. Wichtig dabei ist, dass die Erweiterungen selbst dafür verantwortlich sind, diese Prozesse sauber zu implementieren, denn ein Überprüfen der Daten findet nicht statt. Eine Einschränkung hierbei ist, dass die Daten der Erweiterungen das Interface *Serializable* implementieren müssen. Eine weitere Aufgabe dieser Klasse ist das Starten und Verwalten der Analysemethoden, welche auf dem Interface *Tool* basieren.

### 3.1.8 Erweiterung der Deserialisierung für mehr Flexibilität

Die von Java verwendete Methode der Serialisierung und Deserialisierung bringt einige Einschränkungen mit sich. Eine dieser Einschränkungen ist, dass bei der Serialisierung verwendete Klassen, die nach Änderungen an der Software nicht mehr benötigt werden, nicht gelöscht werden können. Im Fall von MonaLisa betrifft dies also alle Klassen, welche in der Klasse *Project* verwendet werden. Ein Löschen von Klassen führt zu einer *ClassNotFoundException* beim Deserialisieren eines *Project* Objekts, da die Datei der entsprechenden Klasse nicht mehr vorhanden ist. Diese wird jedoch benötigt, um die gespeicherten Daten korrekt zu interpretieren. Um diesen Fall abzufangen, wurde die Java Klasse *java.io.ObjectInputStream* überschrieben. Die so entstandene Klasse *MonaLisaObjectInputStream* gibt in Falle einer *ClassNotFoundException* eine Platzhalterklasse ohne Inhalt zurück und die Deserialisierung gelingt ohne Fehler.

Außerdem erzeugt jede Veränderung der Attribute einer Klasse ein Problem mit der Abwärtskompatibilität. Zur eindeutigen Identifikation einer Klasse wird ein Hashwert, basierend auf deren Quelltext, berechnet. Der Hashwert dient neben der Identifikation auch zur Unterscheidung verschiedener Versionen derselben Klasse, da durch das Hinzufügen neuer Funktionen oder Attribute der Hashwert verändert wird. Wird nun versucht, eine solche ältere Version einer Klasse zu deserialisieren, stellt Java dies anhand des Hashwertes fest und bricht den Vorgang ab. Um dies zu verhindern, ist es in Java möglich, einer Klasse ein Attribut mit dem Namen *serialVersionUID* hinzuzufügen und in diesem einen manuellen Hashwert festzulegen. Ist dieses Attribut vorhanden, berechnet Java den Hashwert beim Deserialisieren nicht neu, sondern verwendet den vorhandenen Wert. Wird der Klasse nun ein neues Attribut hinzugefügt, so wird der Vorgang des Deserialisierens nicht mehr abgebrochen. Neue Attribute, welche nicht im abgespeicherten Objekt vorhanden sind, werden auf den Wert *null* gesetzt. Dies führt jedoch zu weiteren Problemen, da ein Zugriff auf dieses Attribut eine *NullPointerException* auslösen kann. Um eine Abwärtskompatibilität zu gewährleisten, kann der Entwickler über die Funktion *readObject()* in den Prozess der Deserialiserung eingreifen und solch neue Attribute mit einem Standartwert initialisieren. Dieses Vorgehen wurde bei der Entwicklung von MonaLisa notwendig, da vorhandene Klassen erweitert und verändert wurden, so zum Beispiel in der Klasse *Project* selbst, da hier nun ein Attribut vom Typ *Synchronizer* existiert.

Das Umbenennen von Attributen oder Klassen, das Ändern des Typs eines Attributes oder das Verschieben einer Klasse in ein anderes *Package* ist jedoch nicht möglich. Diese Fälle können nicht mit der *readObject()*-Funktion abgefangen werden und auch nicht ohne Weiteres durch Eingreifen in der *java.io.ObjectInputStream*-Klasse gelöst werden. Um diese Probleme zu lösen, wurden im Laufe der Zeit verschiedene Lösungsansätze entwickelt. Einige dieser Methoden wurden von Markus Hildebrand im Rahmen einer Masterarbeit (Hildebrand, 2015) zu der Java-Bibliothek *AutoSerial* zusammengefügt, die mit Hilfe von Annotationen an Klassen und Attributen eine Abwärtskompatibilität gewährleisten soll.

Mit dieser Bibliothek ist es nicht mehr nötig, in die *java.io.ObjectInputStream*-Klasse einzugreifen oder die *readObject()*-Funktion zu verwenden. Letztere existiert in den Klassen nicht mehr, da das Interface *Serializable* nicht mehr verwendet wird. In einer Testumgebung konnte diese Methode erfolgreich angewendet werden. Bei dem Versuch, diese Methode fest in MonaLisa zu integrieren, traten jedoch größere Probleme auf. Diese konnten letztendlich auf Fehler in einer der verwendeten externen Bibliotheken zurückgeführt werden.

Der große Vorteil der Serialisierung ist, dass kein eigenes Dateiformat entwickelt werden muss, was bei einem komplexen Objekt wie der *Project*-Klasse ebenfalls komplex wäre. Weiter muss sich der Entwickler nicht um das Erzeugen der Datei selbst kümmern, dies wird von Java automatisch erledigt. Die erzeugte Datei ist im Falle einer binären Datei zudem wesentlich kleiner als bei einem ASCII-basierten Format oder gar ein XML-basierten Format, was auch das Deserialisieren beschleunigt.

Für die weitere Entwicklung von MonaLisa ist längerfristig die Entwicklung eines eigenen Dateiformats zu empfehlen. Die oben gezeigten Einschränkungen der Serialisierung verhindern eine flexiblere Anpassung der vorhandenen Klassen. Auch der Ansatz von *AutoSerial* hebt nicht alle diese Einschränkungen auf. Ein eigenes Dateiformat zu entwickeln erfordert einen großen Zeitaufwand, nicht nur für die Entwicklung selbst, sondern auch für die Umstellung von MonaLisa auf dieses. Solch ein Format muss flexibel genug sein, um auf Änderungen der zu speichernden Daten reagieren zu können, aber auch unabhängig von der internen Speicherungen dieser sein. Falls eine Erweiterung von MonaLisa um neue PN-Varianten geschieht, wird die Entwicklung eines eigene Dateiformates dringend notwendig. Die hierzu nötigen Anpassungen an vorhandenen Klassen würden zu großen Problemen bei der Serialisierung führen und große Kompromisse bei der weiteren Entwicklung fordern.

### 3.1.9 Der Import und Export externer Dateiformate

Es existieren eine Vielzahl von externen Dateiformaten zur Speicherung biologischer Modelle. Um diese mit MonaLisa nutzen zu können, wurde der Import verschiedener externer Dateiformate implementiert sowie der Export in diese. Somit wird ein Austausch mit anderen Programmen oder Datenbanken ermöglicht. Eine Übersicht der unterstützten Dateiformate gibt Tabelle 3.1. Zu den unterstützten Dateiformaten zählen unter anderem SBML, PNML, DAT und PNT. Beim Import von SBML-Dateien ist das Lesen jeder Version möglich, für den Export kann zwischen der Version (Level 1) oder der Version 2 (Level 5) gewählt werden. Hierzu wird die offizielle Java-Bibliothek jSBML (Rodriguez et al., 2015) verwendet. Für PNML existiert weder eine solche Bibliothek noch eine ausreichende frei verfügbare Dokumentation, weshalb es hier zu Einschränkungen kommt. Unterstützt werden PNML-Dateien, welche mit der Software PIPE (Dingle et al., 2009) erstellt wurden. Jedoch gibt es zwischen den Hauptversionen dieser Software Abweichungen im Dateiformat, sodass für diese jeweils eine eigene Importfunktion zur Verfügung

steht. Das *Plain*-Dateiformat beinhaltet eine Liste aller im PN vorhanden Reaktionen in der Notation chemischer Reaktionen. Details zu diesem Format sind im Anhang zu finden (siehe Abschnitt 5.1).

| Dateiformat | Import | Export |
|---|---|---|
| SBML | ✓ | ✓ |
| PNT | ✓ | ✓ |
| PNML | ✓ | ✓ |
| DAT | ✓ | ✓ |
| KGML | ✓ | – |
| SPPED | ✓ | – |
| APNN | – | ✓ |
| Plain | – | ✓ |

**Tabelle 3.1:** Die Tabelle führt alle von MonaLisa unterstützten Dateiformate auf und zeigt, ob diese nur für den Export, den Import oder beide Wege zur Verfügung stehen.

Durch die Unterstützung der genannten externen Dateiformate wird ein Austausch der mit MonaLisa erstellten Modelle mit zahlreichen anderen Programmen ermöglicht. Für eine künftige Weiterentwicklung von MonaLisa wäre eine Implementierung zur Unterstützung weiterer Dateiformate interessant. Mögliche Kandidaten hierzu wären die Formate SBGN (Novere et al., 2009) und BioPax (Demir et al., 2010). Daneben kann die Unterstützung von SBML weiter ausgebaut werden. Hier könnte zum Beispiel die Implementation für das SBML *Packages qual*, eine Variante von SBML speziell zur Abbildung qualitativer Modelle, erfolgen. Die Unterstützung des *Packages layout* würde das Speichern der räumlichen Koordinaten in einer SBML-Datei ermöglichen. Zu bedenken ist hierbei jedoch, dass Anwendungen, die solch erweiterte SBML-Dateien verarbeiten sollen, diese *Packages* ebenfalls unterstützen müssen. Zur Zeit existiert auch noch nicht für jedes *Package* eine Unterstützung in der jSBML-Bibliothek.

## 3.2 Die grafische Oberfläche von MonaLisa

MonaLisa besteht aus mehreren grafischen Komponenten, die jeweils einem bestimmten Zweck dienen. Die analytische Komponente dient zur grundlegenden Analyse der PN, wie zum Beispiel zur Berechnung der T-Invarianten. Daneben existieren zwei Visualisierungskomponenten, der *NetViewer* und der *TreeViewer*. Im *NetViewer* findet die Modellierung und Visualisierung des PN und die Projektion der Analyseresultate auf dieses statt. Der *TreeViewer* dient zur Visualisierung der T-Cluster und deren hierarchischen Bäume. Die einzelnen Komponenten und deren Funktionen werden in den folgenden Abschnitten näher beschrieben. Eine vollständige Übersicht über alle Funktionen und deren Beschreibung ist im Internet verfügbar (ML-Dokumentation, 2016), weshalb auf eine solche detaillierte Beschreibung in dieser Arbeit verzichtet wird.

## 3.3 Die analytische Komponente

TINA (Thormann et al., 2009) bildet die Grundlage für MONALISA. Diese Software stellte den Import und Export einiger Dateiformate zur Verfügung, wandelte diese in eine interne PN-Repräsentation um und stellte verschiedene Analysemethoden zur Verfügung. Dieses Grundgerüst wurde im Zuge dieser Arbeit um weitere Analysemethoden und Funktionen erweitert und bildet die analytische Komponente von MONALISA. In Abbildung 3.7 wird eine Übersicht über die grafische Oberfläche von MONALISA gegeben. Gezeigt wird neben dem Startbildschirm (links) auch die Oberfläche zur Analyse der PN (rechts). Auf diesem Niveau ist das Anlegen eines neuen Projektes, das Öffnen eines bereits existierenden Projektes sowie das Importieren eines externen Dateiformates möglich. Ist einer dieser drei Wege gewählt, stehen die Analysemethoden der analytischen Komponente zur Verfügung, wie zum Beispiel die Berechnung der T-Invarianten oder MCT-Sets. Der Zugriff auf den *NetViewer* und den *TreeViewer* erfolgt über die entsprechenden Schaltflächen. Weiter

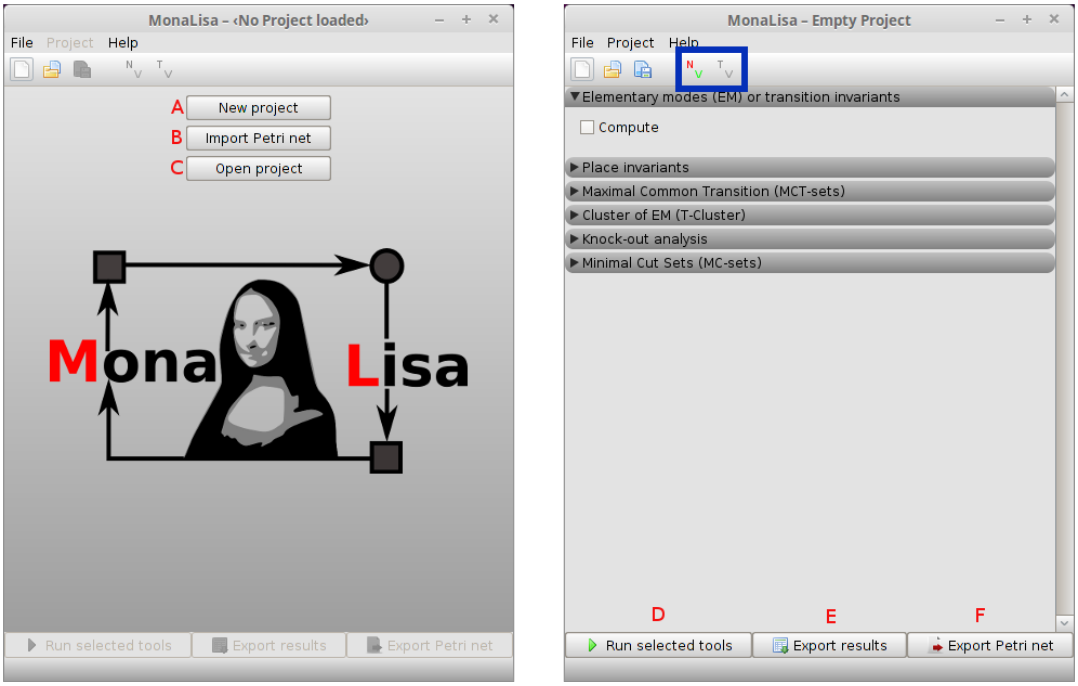


**Abbildung 3.7: Links:** Der Startbildschirm von MONALISA. Von diesem aus ist es möglich, (A) ein neues, leeres Projekt anzulegen, (B) ein PN über ein externes Dateiformat zu importieren oder (C) ein MONALISA-Projekt zu laden. Diese Funktionen sind sowohl über die markierten Schaltflächen verfügbar, als auch über den Menüpunkt *File* in der oberen Menüleiste.
**Rechts:** Übersicht über die verfügbaren Analysemethoden. Die Menüs können aufgeklappt werden, womit die Einstellungsmöglichkeiten der einzelnen Methoden sichtbar werden. Am unteren Rand befinden sich Schaltflächen zum (D) Starten der ausgewählten Analysemethoden, (E) zum Export der Resultate dieser Methoden und (F) zum Export des PN in externe Dateiformate. Diese Funktionen sind ebenfalls über den Menüpunkt *File* und *Project* verfügbar. Der *NetViewer* (NV – Editieren und Betrachten des PN) und der *TreeViewer* (TV – Visualisierung der T-Cluster) können über die Schaltflächen im blau umrandeten Bereich geöffnet werden. Die Schaltfläche für den *TreeViewer* ist hier deaktiviert, da noch keine entsprechenenden Resultate vorliegen.

steht ein Export des PN in ein externes Dateiformat oder der Resultate in ASCII Dateien zur Verfügung. Über den Menüpunkt *File* im oberen Bereich ist der Zugriff auf einen Petrinetz-Konverter möglich, mit welchem automatisch Dateien von einen externen Dateiformat in ein anderes umgewandelt werden können. Eine nähere Beschreibung dieser Funktionen folgt in den nächsten Abschnitten.

### 3.3.1 Der Petrinetz-Konverter

Der Petrinetz-Konverter ist eine Funktion von MonaLisa, mit der sich automatisch viele Dateien auf einmal von einem externen Dateiformat in ein anderes umwandeln lassen. Zur Verfügung stehen alle von MonaLisa unterstützten externen Dateiformate. Der Konverter ist über den Menüpunkt *File* → *Converter* erreichbar. Abbildung 3.8 zeigt das Menü des Petrinetz-Konverters. Vor dem Start muss sowohl ein Quell- als auch ein Zielverzeichnis festgelegt werden, und ebenso das Quell- und Zielformat.

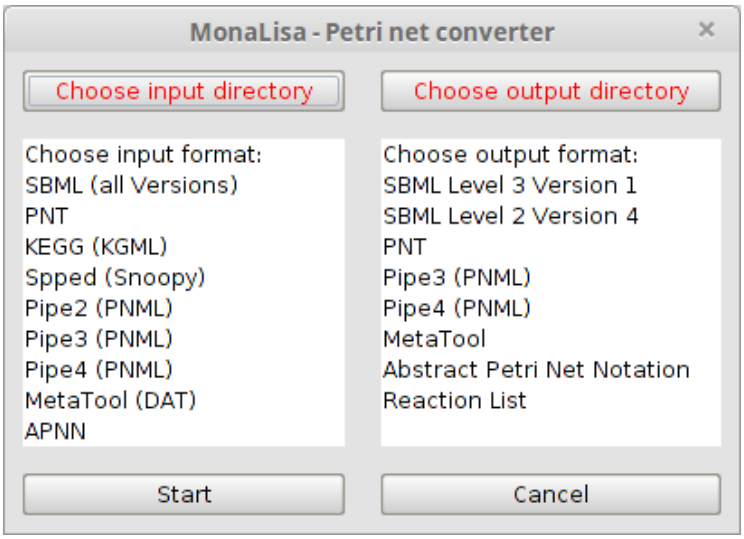


**Abbildung 3.8:** Die Abbildung zeigt das Menü des Petrinetz-Konverters. Mit ihm können beliebig viele Dateien eines Quellverzeichnisses und Dateiformats (links) in ein anderes Dateiformat (rechts) konvertiert werden. Diese werden im ausgewählten Zielverzeichnis gespeichert. Zur Auswahl stehen alle von MonaLisa unterstützten Dateiformate zum Import und Export.

### 3.3.2 Transitions-Invarianten

Die Berechnung der T-Invarianten erfolgt nicht in MonaLisa selbst, sondern wird von einem externen C-Programm übernommen. Dieses wurde von Jörg Ackermann implementiert und verwendet zum Lösen des Gleichungssystems eine Implementierung der Fourier-Motzkin-Elimination (Fourier, 1826; Colom und Silva, 1991). Eine nähere Beschreibung der Implementierung ist in Koch und Ackermann (2013) zu finden. Dieses Vorgehen wurde gewählt, da eine Implementierung in Java eine viel höhere Laufzeit zur Folge hätte. Für die Berechnung der T-Invarianten wird das PN in eine PNT-Datei exportiert, welches dann vom externen Programm eingelesen wird. Dieses führt nun die Berechnung der

T-Invarianten durch und übergibt diese an MonaLisa, diesmal durch ein internes Dateiformat. Der obere Teil der Abbildung 3.9 zeigt das Menü, welches zum Berechnen der T-Invarianten dient. In diesem Menü wird, sobald die T-Invarianten vorliegen, auch angezeigt, ob das PN die *CTI*-Eigenschaft besitzt. Die T-Invarianten können nun in eine ASCII-Datei exportiert (siehe Abschnitt 5.2) oder im *NetViewer* betrachtet werden. In MonaLisa werden T-Invarianten als *Emelemtary Modes* (EM) (Schuster und Hilgetag, 1994; Koch et al., 2005) bezeichnet, da diese Bezeichnung für Biologen geläufiger ist.

Die T-Invarianten eines PN zählen zu dessen strukturellen Eigenschaften, sie sind also unabhängig von einer Markierung und hängen nur von der Topologie des PN ab (Murata, 1989). Jedoch ist, durch die Gewichtung der Kanten in einem PN, in den Lösungsvektoren der T-Invarianten auch eine quantitative Komponente enthalten. Bei der Untersuchung eines Systems ist die Untersuchung seiner strukturellen Invarianten ein möglicher Startpunkt für weitere Untersuchungen (Zevedei-Oancea und Schuster, 2003). Sie helfen das Verhalten des Systems zu analysieren und funktionelle Einheiten innerhalb des Systems zu identifizieren. Eine T-Invariante bildet eine solche funktionelle Einheit, so ist sie zum Beispiel im Kontext eines metabolischen System eine Menge von Enzymen, welche sich im Fließgleichgewicht befinden. Die erfolgreiche Verwendung von T-Invarianten zur Validierung eines PN wurde von Koch et al. (2005) am Beispiel der Verarbeitung von Saccharose in der Wurzel der Kartoffelpflanze gezeigt. Auch für die Validierung eines Signaltransduktionsweges (Heiner et al., 2004) oder die Modellierung einer Genregulation (Grafahrend-Belau et al., 2008; Grunwald et al., 2008) können T-Invarianten verwendet werden.

### 3.3.3 Platz-Invarianten

Für die Berechnung der P-Invarianten wird dasselbe C-Programm wie zur Berechnung der T-Invarianten verwendet. Die Transponierung der Adjazenzmatrix wird jedoch nicht im externen C-Programm durchgeführt, sondern findet vor dem Export in die PNT-Datei statt. In dieser werden Plätze und Transitionen vertauscht, was so zu einer Berechnung der P-Invarianten führt. Im unteren Teil der Abbildung 3.9 wird das Menü zur Berechnung der P-Invarianten gezeigt. Für diese ist ebenfalls ein Export in eine ASCII-Datei (siehe Abschnitt 5.2) und die Darstellung im *NetViewer* möglich.

Wie die T-Invarianten zählen die P-Invarianten ebenfalls zu den strukturellen Eigenschaften eines PN. Im Kontext eines metabolischen Modells stellt eine P-Invariante eine Stoffkonservierung dar (Zevedei-Oancea und Schuster, 2003). Die Anwendung von P-Invarianten zur Analyse und Validierung eines metabolischen Modells wurde von Koch und Heiner (2008) und Heiner et al. (2004) gezeigt. MonaLisa stellt mit den T- und P-Invarianten zwei grundlegende strukturelle Analysemethoden zur Verfügung. Diese ermöglichen ihrerseits eine Analyse und Validierung des PN, bilden aber auch die Grundlage für viele weitere Analysemethoden.

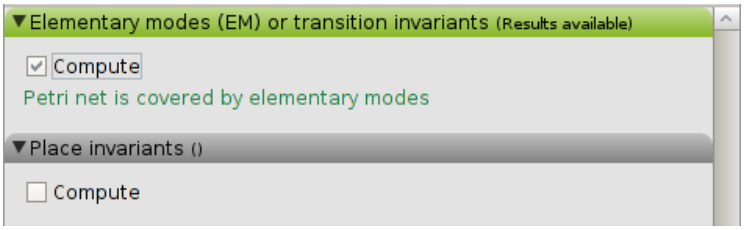


**Abbildung 3.9:** Die Abbildung zeigt das Menü, welches zur Berechnung der T-Invarianten (oberer Teil) und P-Varianten (unterer Teil) dient. Im Teil der T-Invarianten wird nach deren Berechnung angezeigt, ob das PN die *CTI*-Eigenschaft erfüllt oder nicht. Die grüne Unterlegung der Kopfzeile zeigt an, dass diese schon berechnet wurden und Resultate zur Verfügung stehen. Dies ist für die P-Invarianten nicht der Fall, weshalb hier die Kopfzeile grau unterlegt ist. Wird das Feld *Compute* ausgewählt, kann die Berechnung dieser Analysemethode über die Schaltfläche *Run selcted tools* gestartet werden. Diese befindet sich am unteren Ende des gesamten, in Abbildung 3.7 gezeigten, Menüs.

### 3.3.4 *Maximal Common Transition Sets*

In Abbildung 3.10 ist das Menü zur Berechnung der *Maximal Common Transition Sets* (MCT-Sets) gezeigt. Als Basis der MCT-Sets können die Supportvektoren, *Support-oriented*, oder die Parikh-Vektoren, *Occurence-oriented*, der T-Invarianten dienen. In beiden Fällen existiert die Möglichkeit, triviale T-Invarianten in die Berechnung mit einzubeziehen oder nicht. Die berechneten MCT-Sets können in eine ASCII-Datei exportiert (siehe Abschnitt 5.3) oder im *NetViewer* dargestellt werden.

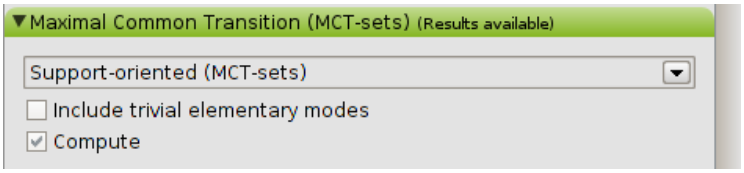


**Abbildung 3.10:** Die Abbildung zeigt das Menü, welches zur Berechnung der MCT-Sets dient. Über die Auswahlliste kann die Variante zur Berechnung der MCT-Sets gewählt werden. Zur Auswahl stehen die Varianten *Support-oriented* oder *Occurence-oriented*. Über das Feld *Include trivial elementary modes* können triviale T-Invarianten mit in die Berechnung einbezogen werden.

Die Komplexität eines PN-Modells kann schnell zunehmen, wodurch auch die Anzahl der T-Invarianten stark zunehmen kann (Klamt und Stelling, 2002). Je größer die Anzahl dieser wird, desto aufwendiger und schwieriger wird deren biologische Interpretation. Aus diesem Grund werden Methoden benötigt, welche die große Menge der T-Invarianten reduzieren können und trotzdem funktionelle, biologisch interpretierbare Einheiten im PN finden können. Eine solche Methode wird mit den MCT-Sets bereitgestellt. MCT-Sets können als Bausteine des PN mit eigener biologischer Bedeutung interpretiert werden, wobei zu beachten ist, dass MCT-Sets immer disjunkte Mengen an Transitionen darstellen. Ein solcher Baustein kann eine Signaleinheit sein, eine Menge von Genen, welche derselben Regulierung unterliegen oder eine Menge von enzymatischen Reaktionen, welche immer zusammen stattfinden. So wurde mit Hilfe der MCT-Sets zum Beispiel

der Paarungspheromon-Antwort-Signalweg in *Saccharomyces cerevisiae* in verschiedene funktionale Module unterteilt (Sackmann et al., 2006), ebenso der Kernmetabolismus von *Arabidopsis thaliana* (Nöthen, 2014).

### 3.3.5 T-Cluster

Zur Berechnung der T-Cluster wird eine Java-Bibliothek von Heiko Giese verwendet, die ursprünglich für die Software NOVA (Giese et al., 2015) entwickelt wurde. In Abbildung 3.10 wird das Menü zur Berechnung dieser gezeigt. Zur Auswahl stehen drei verschiedene Distanzmaße, der Tanimoto-Index, der M-Koeffizient und die *Sum of Differences*. In Kombination mit diesen Distanzmaßen und einem der Clusterverfahren, UPGMA, WPGMA, *Single linkage* oder *Compelte linkage*, können die T-Cluster berechnet werden. Für diese Berechnung stehen sowohl ein Grenzwert, bei dem die Baumkonstruktion beendet wird, als auch die Einbeziehung der trivialen T-Invarianten zur Verfügung. Die erzeugten T-Cluster können anschließend im *TreeViewer* (siehe Abschnitt 3.5) betrachtet werden.

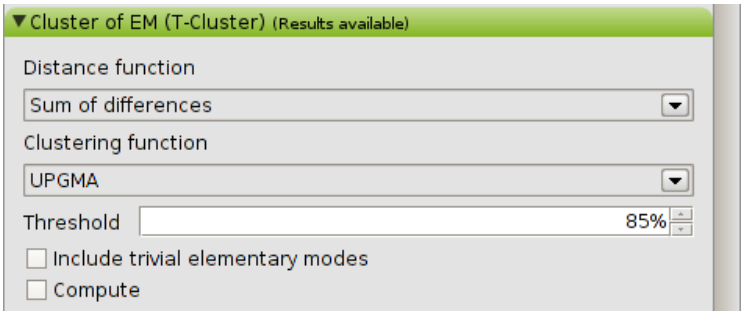


**Abbildung 3.11:** Die Abbildung zeigt das Menü, das zur Berechnung der T-Cluster dient. Über die Auswahllisten kann aus verschiedenen Distanzfunktionen und Clusterverfahren gewählt werden. Über das Feld *Treshold* kann die Höhe bestimmt werden, auf der der Clusterbaum abgeschnitten wird. Über das Feld *Include trivial elementary modes* können die trivialen T-Invarianten in die Berechnung mit einbezogen werden.

Die T-Cluster stellen eine weitere Methode dar, um große Mengen an T-Invarianten zu reduzieren. Die hier gebildeten Cluster von T-Invarianten können als biologisch funktionelle Module interpretiert werden. Im Gegensatz zu den MCT-Sets wird dabei die Überlappung von T-Invarianten zugelassen, was ein weniger stringentes Kriterium darstellt. T-Cluster wurden erfolgreich angewendet, um zum Beispiel ein Modell des Paarungspheromon-Antwort-Signalweges in *Saccharomyces cerevisiae* oder der Genregulierung der Duchenne Muskeldystrophie auf funktionale Module zu untersuchen (Grafahrend-Belau et al., 2008).

Mit den zwei Analysemethoden, den MCT-Sets und den T-Clustern, stellt MonaLisa Methoden zur Reduzierung der T-Invarianten bereit. So können selbst komplexe PN mit einer sehr großen Anzahl an T-Invarianten interpretiert werden und in funktionelle Einheiten unterteilt werden. Wenn das PN jedoch so komplex ist, dass eine Berechnung aller T-Invarianten nicht mehr möglich ist, können auch diese beiden Analysemethoden nicht

angewendet werden. Ein Ansatz hier ist, nicht die T-Invarianten zu reduzieren, sondern das ganze PN. Eine solche Methode wurde in Ackermann et al. (2012) vorgestellt, wobei die Reduzierung des PN nach solchen Regeln erfolgt, die die *CTI*-Eigenschaft des PN konservieren. Eine solche Reduzierung kann die Berechnung der T-Invarianten ermöglichen, so auch gezeigt in Nöthen (2014). Im Zuge der Weiterentwicklung von MonaLisa wäre die Ergänzung um diese Methode ein nächster Schritt.

### 3.3.6 Knock-out-Analyse

In Abbildung 3.10 wird das Menü zur Durchführung der Knock-out-Analysen gezeigt. Es stehen verschiedene Verfahren zur Verfügung. Zu oberst gibt es automatisierte Methoden, um jeden einzelnen Platz oder jede einzelne Transition oder alle Paare dieser auszuschalten. Es kann aber auch eine bestimmte Auswahl an Transitionen oder Reaktionen getroffen werden, für die eine Knock-out-Analyse durchgeführt werden soll. Die Resultate dieser Analysen können in eine ASCII-Datei exportiert werden (siehe Abschnitt 5.4). Eine Darstellung der Knock-out-Analysen im *NetViewer* erfolgt über ein gesondertes Menü dort und basiert nicht auf den hier erzeugten Daten (siehe Abschnitt 3.4.1). In MonaLisa werden Plätze mit dem Begriff *Species* und Transitionen mit dem Begriff *Reactions* benannt, um eine bessere Terminologie für die Systembiologie zu erreichen.

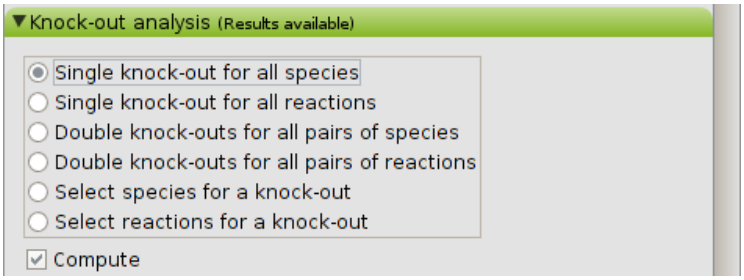


**Abbildung 3.12:** Die Abbildung zeigt das Menü, das zum Durchführen der Knock-out-Analysen dient. Es kann ausgewählt werden, welche der verschiedenen Knock-out-Analysen gestartet wird. Es steht sowohl das automatisierte Ausschalten einzelner Transitionen oder Plätze, in MonaLisa mit *Species* benannt, als auch aller Paare dieser zur Verfügung. Weiter ist das Ausschhalten einer bestimmten Auswahl an Plätzen oder Transitionen möglich. Das Menü zur Auswahl dieser öffnet sich nach dem Benutzen der Schaltfläche *Run selected tools*.

Für die Durchführung einer Knock-out-Analyse werden die T-Invarianten bzw. P-Invarianten des PN benötigt. Anders als jedoch bei den MCT-Sets oder den T-Clustern dient die Knock-out-Analyse nicht der Reduzierung der Invarianten. Ziel dieser Analyse ist, zu untersuchen welchen Einfluss auf das Fließgleichgewicht oder die Stoffkonservierung das Ausschalten eines Platzes oder einer Reaktion hat. So können zum Beispiel alternative Wege in einem metabolischen Modell oder aber das Fehlen dieser aufgezeigt werden. Eine Anwendung dieser Analysemethode wird in Grunwald et al. (2008) anhand eines PN-Modells der Duchenne Muskeldystrophie gezeigt.

### 3.3.7 *Minimal Cut Sets*

In Abbildung 3.13 wird das Menü zur Berechnung der *Minimal Cut Sets* (MCS) gezeigt. In diesem kann die Transition, welche deaktiviert werden soll, ausgewählt und die maximale Größe der zu berechnenden MCS festgelegt werden. Diese können anschließend in eine ASCII-Datei exportiert (siehe Abschnitt 5.5) oder im *NetViewer* dargestellt werden.


**Abbildung 3.13:** Die Abbildung zeigt das Menü zur Berechnung der MCS. Über die Auswahllisten kann die zu deaktivierende Transition ausgewählt werden. Darunter kann die maximale Größe der berechneten MCS festgelegt werden.

Die Fragestellung der MCS ist genau umgekehrt zu der Fragestellung der Knock-out-Analyse. Ziel der MCS ist das Finden einer Menge von Transitionen, durch dessen Deaktivierung eine gegebene Transition ausgeschaltet wird. Wie von den Autoren Klamt und Gilles (2004) beschrieben, ist die Einsatzmöglichkeit der MCS sehr groß. Mit Hilfe derer lässt sich zum Beispiel die Fragilität und die Robustheit eines PN untersuchen. Aber auch die Vorhersage des Phänotyps eines Mutanten oder die Identifizierung möglicher Ziele eines Wirkstoffes sind mit MCS möglich. In derselben Veröffentlichung wird auch ein Fragilitäts-Koeffizient $F$ für jede Transition des PN vorgestellt. Dieser ist das Reziproke der durchschnittlichen Größe aller MCS, in der eine Reaktion vorkommt. Die Analyse der MCS in MonaLisa könnte durch die Implementierung dieses Fragilitäts-Koeffizienten erweitert werden.

## 3.4 Der *NetViewer*

Im *NetViewer* findet die Modellierung und Visualisierung des PN und die Projektion der Resultate aus den Analysemethoden auf dieses statt. Die Abbildung 3.14 gibt eine Übersicht über die Oberfläche des *NetViewer*. Dieser besteht aus zwei Grundelementen: Links ist der Bereich, in dem die Modellierung stattfindet, der PN-Editor, rechts befindet sich die *ToolBar*, in der sich die Funktionalitäten des *NetViewer* befinden. Die *ToolBar* selbst hat verschiedenste Karteireiter, die jeweils zu einer bestimmten Funktionalität gehören. Es gibt drei Karteireiter, die zum Kern des *NetViewer* gehören: *Control*, *Analysis* und *SearchBar*. Die restlichen Karteireiter gehören zu ergänzenden Analysemethoden des *NetViewer*. Im Folgenden wird sowohl auf den PN-Editor als auch auf die einzelnen Karteireiter näher eingegangen.

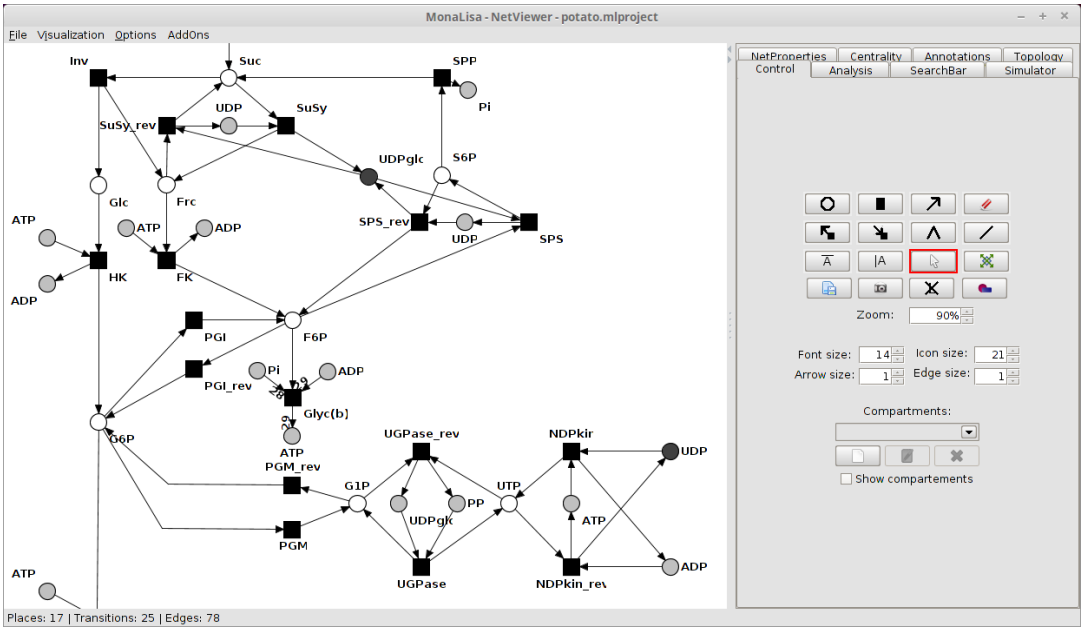


**Abbildung 3.14:** Die Abbildung gibt eine Übersicht über die grafische Oberfläche des *NetViewer*. **Links:** Der Bereich des PN-Editors. Hier wird das PN modelliert und angezeigt. Transitionen werden durch Rechtecke repräsentiert, Plätze durch Kreise. Das hier gezeigte PN ist ein Ausschnitt aus einem Modell der Verarbeitung von Saccharose in der Wurzel der Kartoffelpflanze (Koch et al., 2005).
**Rechts:** Die *ToolBar*. Sie besteht aus mehreren Karteireitern, die alle zu einer bestimmten Funktionalität des *NetViewer* gehören. Gezeigt wird der Karteireiter *Control*, mit dem die Modellierung des PN gesteuert wird.

### 3.4.1 Der PN-Editor

Im PN-Editor kann mit der Maus und der *ToolBar* (siehe Abschnitt 3.4.2) ein PN modelliert werden. Der linke Teil der Abbildung 3.15 zeigt diesen. In ihm können Plätze, dargestellt durch Kreise, Transitionen, dargestellt durch Rechtecke, und Kanten zwischen diesen erzeugt und angeordnet werden. Über das Kontextmenü der einzelnen Elemente können deren Eigenschaften angepasst und weitere Funktionen erreicht werden. Das Menü zum Anpassen der Eigenschaften eines Knotens ist im rechten Teil der Abbildung gezeigt. Im oberen Teil dieses Menüs ist gekennzeichnet, um was für einen Knotentyp es sich handelt und welcher Knoten gerade bearbeitet wird. Im Bereich *Shape* kann die Form des Knotens bestimmt werden. Gewählt werden kann zwischen einem Kreis und regelmäßigen Vielecken von bis zu sieben Ecken. Beispiele hierfür sind im linken Teil der Abbildung die Transitionen *SucTrans* und *eSuc*. Der Bereich *Style* erlaubt das Ändern des Namens, der Anzahl an Marken eines Platzes, der Farbe des Knotens und der Farbe seiner Umrandung. Die Position des Namens am Knoten selbst kann im Bereich *Label position* bestimmt werden. Weitere Optionen sind das Zuweisen des Knotens zu einem Zellkompartiment (siehe Abschnitt 3.4.2) und das Hinterlegen einer Notiz zu diesem Knoten. Diese wird als Kurzinformation angezeigt, wenn der Mauszeiger auf diesem Knoten

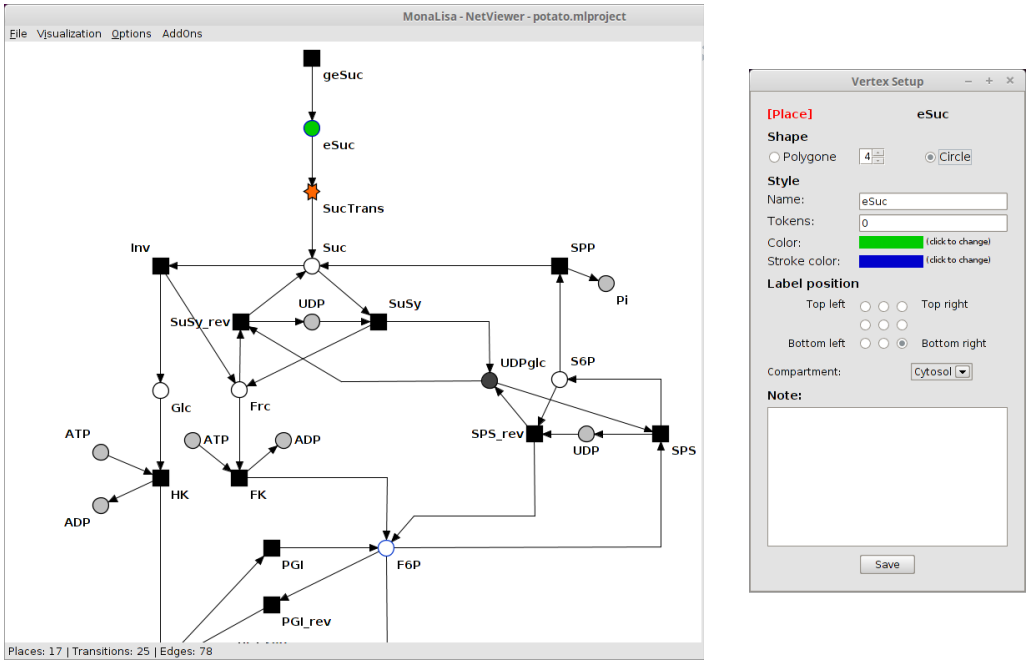


**Abbildung 3.15:** Die Abbildung zeigt den PN-Editor (links) und das Menü zum Anpassen der Plätze und Transitionen (rechts)
**Links:** Der Bereich des PN-Editors. Hier können Plätze, in Form von Kreisen, Transitionen, in Form von Rechtecken, und Kanten zwischen diesen angelegt werden. Standardmäßig haben Plätze keine und Transitionen eine schwarze Füllung. Wie am Platz *eSuc* und an der Transition *SucTrans* zu erkennen ist, kann diese Eigenschaft, wie auch die Form, für jeden Knoten angepasst werden. Das Menü hierzu ist über das Kontextmenü der Knoten und Kanten verfügbar und wird im rechten Teil der Abbildung gezeigt. Die grauen Plätze von *ATP* und *ADP* stellen logische Plätze dar. In der Menüleiste stehen weitere Funktionen zur Verfügung. So können zum Beispiel über den Menüeintrag *Visualization* alle Transitionen farblich hervorgehoben werden, die in mindestens einer T-Invariante vorkommen. In der Statusanzeige unter dem PN-Editor werden Informationen über das PN zusammengefasst, zum Beispiel wie viele Plätze, Transitionen und Kanten in diesem vorhanden sind.
**Rechts:** Das Menü zum Festlegen der Eigenschaften der Knoten des PN. Neben der Form, im Bereich *Shape*, kann über dieses Menü der Name des Knotens, im Falle eines Platzes dessen Anzahl an Marken, die Farbe zum Füllen des Knotens und für dessen Umrandung festgelegt werden. Im Bereich *Label position* kann die Position des Namens am Knoten bestimmt werden. Falls für das PN Zellkopartimente angelegt wurden, lässt sich über das Menüelement *Compartement* ein Knoten zu diesen zuordnen. Im unteren Textfeld *Note* kann eine Notiz zu diesem Knoten hinterlegt werden. Dieser wird dann im PN-Editor als Kurzinformation angezeigt, wenn die Maus über diesem Knoten verweilt.

verweilt. Solch eine Kurzinformation kann auch für Kanten hinterlegt werden, deren Farbe und Kantengewicht ebenfalls anpassbar sind. Über das Kontextmenü können zudem logische Plätze erzeugt werden. Ein logischer Platz wird benutzt, um eine bessere visuelle Repräsentation des PN zu schaffen. Er stellt die grafische Kopie eines vorhandenen Platzes dar, welche nur mit einem Teil der ursprünglichen Transitionen verbunden ist. In der eigentlichen Struktur des PN wird diese Kopie ignoriert und die ursprüngliche Topologie beibehalten. Logische Plätze werden im PN-Editor standardmäßig in grau angezeigt, wie in der Abbildung zum Beispiel die Knoten von ATP und ADP. Für einige Funktionen

stehen im Kontextmenü automatisierende Funktionen zur Verfügung. So kann zum Beispiel automatisch eine reversible Transition erzeugt werden. In der oberen Menüleiste des PN-Editors (linker Teil der Abbildung) stehen über die einzelnen Menüpunkte weitere Funktionen zur Verfügung. Unter *File* kann das aktuelle Projekt gespeichert werden. Im Menüpunkt *Visualization* kann unter anderem die Einfärbung des PN rückgängig gemacht werden, alle Knotenbeschriftungen ausgeblendet oder das PN im *NetViewer* zentriert werden. Zudem steht hier eine Funktion zur Verfügung, alle Transitionen, welche mindestens in einer T-Invariante vorkommen, farblich hervorzuheben. So kann leicht überprüft werden, ob und welche Transitionen nicht von T-Invarianten überdeckt sind. Der Menüpunkt *Options* öffnet ein Menü zum Festlegen der Standardfarben für die farbliche Hervorhebung der Resultate der einzelnen Analysemethoden (siehe Abschnitt 3.4.3). Dieses Menü wird in Abbildung 3.16 gezeigt.

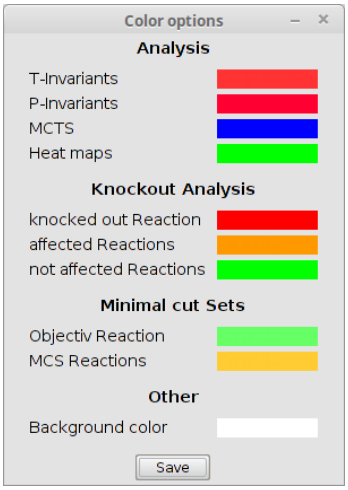


**Abbildung 3.16:** Die Abbildung zeigt das Menü zum Anpassen der Farboptionen des *NetViewer*. Im Bereich *Analysis* werden die Farben zur Einfärbung der Resultate der einzelnen Analysemethoden festgelegt, bis auf die MCS, welche im Bereich *Minimal cut Sets* festgelegt werden. Die Farben für die Knock-out-Analyse werden im Bereich *Knock-out-Analysis* festgelegt. Im untersten Bereich wird die Hintergrundfarbe im *NetViewer* festgelegt.

Für einige Analysemethoden, wie die Knock-out-Analyse oder den MCS, stehen mehrere Optionen zur Verfügung, so zum Beispiel die Farbe der vom Knock-out betroffenen Transitionen oder der nicht betroffen Transitionen. Die Hintergrundfarbe des PN-Editors kann ebenfalls frei gewählt werden. Im Menüpunkt *AddOns* können, bis auf die Karteireiter *Control* und *Analysis*, die einzelnen Karteireiter der *ToolBar* ein- und ausgeblendet werden.

In der Statusanzeige unter dem PN-Editor wird angezeigt, wie viele Plätze, Transitionen und Kanten im aktuellen PN vorhanden sind. Bei der Anzahl der Plätze werden logische Plätze jeweils nur einmal gezählt.

Der PN-Editor bietet eine Vielzahl von Funktionen, nicht nur zur Modellierung sondern auch für einfache Analysen des PN. So dient die Einfärbung aller Transitionen, die in mindestens einer T-Invariante vorkommen, der schnellen Überprüfung der *CTI*-Eigenschaft.

Sind Transitionen nicht von T-Invarianten überdeckt, so sind diese nicht im Fließgleichgewicht, was zu einer Eliminierung oder einer Akkumulation von Stoffen führt. Solche Bereiche des PN sind entweder noch nicht vollständig modelliert oder können weggelassen werden. Diese können durch die Einfärbung schnell identifiziert werden.

Für die künftige Weiterentwicklung könnte der PN-Editor um ein Gitter zur besseren Positionierung der Plätze und Transitionen ergänzt werden. Zur Zeit kann die Größe der Icons und der Beschriftung nur für alle Elemente angepasst werden (siehe Abschnitt 3.4.2). Für die Zukunft wäre denkbar, diese Anpassung für jedes Element einzeln zu ermöglichen. Unabhängig von der Modellierung des eigentlichen PN könnte der PN-Editor um eine Möglichkeit erweitert werden, frei grafische Elemente im Modell zu platzieren, um zum Beispiel eine Zellmembran oder ähnliches darzustellen. Dies würde die Elemente des PN in einen biologischen Kontext bringen und eine bessere Präsentation des Modells für Vorträge oder Veröffentlichungen ermöglichen.

#### Visuelle Knock-out-Analyse

Im *NetViewer* ist ebenfalls eine Knock-out-Analyse möglich. Diese Funktion ist über das Kontextmenü einer Transition verfügbar, falls berechnete T-Invarianten vorhanden sind und bietet mehrere Optionen. Ein Beispiel für solch eine Knock-out-Analyse ist in Abbildung 3.17 zu sehen. Angewendet wurde hier die normale Knock-out-Analyse, welche die gewählten Transitionen rot einfärbt, ebenfalls vom Knock-out betroffene Transitionen orange und die restlichen Transitionen grün einfärbt. Eine andere Variante der Einfärbung für nicht betroffene Transitionen ist eine *Heatmap* auf Grundlage der Anzahl von T-Invarianten, in welcher eine solche Transition nach dem Knock-out noch enthalten ist.

Die visuelle Darstellung der Knock-out-Analyse unterstützt den Modellierer bei der Interpretation der Resultate. Durch die Einfärbung der Transitionen sind sofort die Bereiche des PN zu erkennen, die sich nicht mehr im Fließgleichgewicht befinden.

Im *NetViewer* ist es zur Zeit nur möglich, Transitionen auszuschalten. Die Ergänzung um eine visuelle Knock-out-Analyse für Plätze wäre eine sinnvolle Weiterentwicklung für MonaLisa. Eine alternative Darstellung der Knock-out-Analyse, die *Mauritius Map*, wurde von Grunwald et al. (2008) vorgeschlagen. Die *Mauritius Map* ist eine endlicher, binärer Baum, dessen Knoten Transitionen repräsentieren. Die Ergänzung um die Darstellung solcher *Mauritius Maps* würde die Analyse und Interpretation der Knock-out-Analyse erweitern und verbessern.

### 3.4.2 Der Karteireiter *Control*

Über diesen Karteireiter lässt sich im *NetViewer* das PN modellieren und ermöglicht den Zugriff auf eine Reihe weiterer Funktionen. Für die Modellierung ist der obere Bereich zuständig, in Form einer Sammlung von Schaltflächen. Die erste Reihe dieser ist, von links nach rechts, für das Hinzufügen neuer Plätze und Transitionen, das Ziehen neuer Kanten und das Löschen von Elementen aus dem PN zuständig. Die ersten beiden Schaltflächen

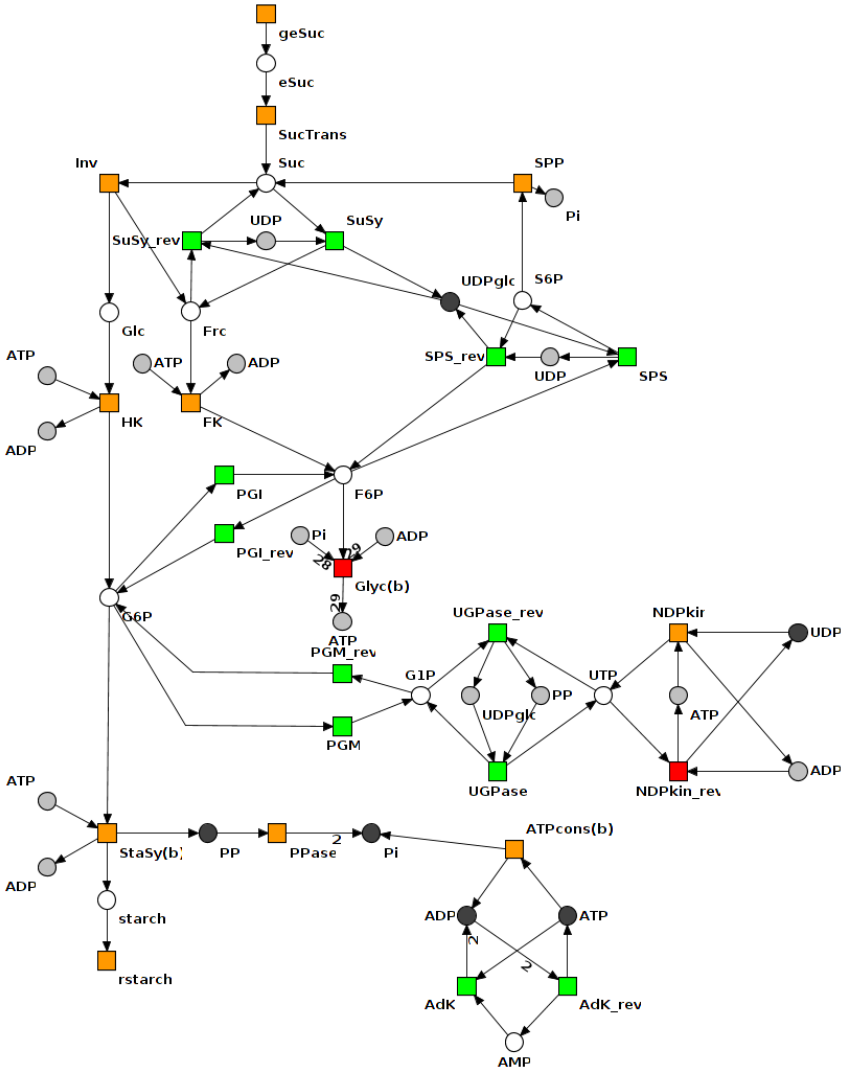


**Abbildung 3.17:** Die Abbildung zeigt eine visuelle Knock-out-Analyse im *NetViewer*. Rot werden die zum Knock-out gewählten Transitionen eingefärbt. Ist eine Transition nun noch immer in mindestens einer T-Invarinate enthalten, so wird diese grün eingefärbt. Ist dies nicht der Fall, so erfolgt eine Einfärbung in orange. Das hier gezeigte PN ist ein Modell der Verarbeitung von Saccharose in der Wurzel der Kartoffelpflanze (Koch et al., 2005).

der zweiten Reihe fügen automatisch einen neuen Vorplatz oder eine neue Vortransition an einen ausgewählten Knoten an und umgekehrt. Über die hinteren beiden Schaltflächen können Knicke in Kanten eingefügt oder wieder entfernt werden. In der dritten Reihe befinden sich Schaltflächen zum horizontalen und vertikalen Ausrichten von Knoten (die ersten beiden von links). Die beiden rechten Schaltflächen in dieser Reihe dienen zum Wechseln zwischen den beiden Mausmodi. Es kann zwischen einer Maus gewählt werden, die das Auswählen und Verschieben der Elemente zulässt und einer Maus, mit der nur das gesamte PN verschoben werden kann. Die vierte Reihe hat, von links nach rechts, die Funktion des Speicherns des aktuellen PN in einer Datei, die aktuelle Ansicht des *NetViewer* in einen Bild zu exportieren, im PNG- oder SVG-Format, die Beschriftung

der Knoten ein- und auszublenden und die Einfärbung aller Knoten zu deaktivieren. Unter diesem Block von Schaltflächen befindet sich ein Menüelement zum Verändern des Zoom-Levels und zur Veränderung der Schrift- und Icongröße sowie der Dicke der Kanten und Größe der Pfeilspitzen am Ende dieser. Die hier gewählten Werte werden für jedes MonaLisa-Projekt getrennt gespeichert. Im unteren Bereich *Compartments* befindet sich das Menü zum Verwalten der Zellkompartimente. Dieses Menü wird im Abschnitt 3.4.2 näher beschrieben.

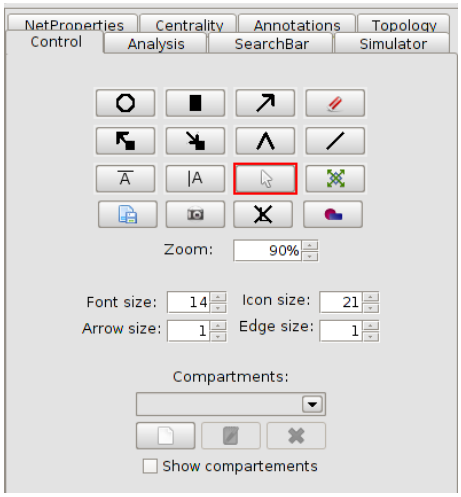


**Abbildung 3.18:** Die Abbildung zeigt den Karteireiter *Control*. Der obere Block von Schaltflächen dient zur Modellierung des PN. Über die einzelnen Schaltflächen können zum Beispiel in der erste Reihe, von links nach rechts, neue Plätze und Transitionen angelegt werden, Kanten gezogen oder Elemente des PN gelöscht werden. Über andere Schaltflächen können zum Beispiel Knoten aneinander ausgerichtet werden oder die aktuelle Ansicht im *NetViewer* als Bild gespeichert werden. Unter diesem Block von Schaltflächen befinden sich Elemente, über die das Zoom-Level im *NetViewer* bestimmt werden kann und Parameter wie die Schrift- oder Icongröße, festgelegt werden können. Im unteren Teil des Karteireiters befindet sich das Menü zum Verwalten der Zellkompartimente.

### Zellkompartimente

Der Karteireiter *Control* stellt die Verwaltung der verschiedenen Zellkompartimente bereit. Das Anlegen eines solchen ist in Abbildung 3.19 zu sehen. Dabei kann ein Name und eine Farbe für das Zellkompartiment festgelegt werden. Die Farbe kann später benutzt werden, um alle Elemente, die diesem Zellkompartiment zugewiesen sind, über die Schaltfläche *Show compartments* in dieser einzufärben. Für eine volle Unterstützung des SBML-Formates sind weitere Informationen über das Zellkompartiment nötig, die über die entsprechenden Optionen angegeben werden können. Dazu zählen die Anzahl der räumlichen Dimensionen, die Größe des Zellkompartiments sowie die Angabe, ob diese konstant ist oder sich ändern kann. Über die Schaltfläche *Edit Annoations* kann die Annotation des Zellkompartments vorgenommen werden. Details hierzu finden sich im Abschnitt 3.4.9.

Die Zuordnung eines Platzes oder einer Transition zu einem bestimmten Zellkompartiment erfolgt über das Menü der Eigenschaften dieser (siehe Abbildung 3.15).

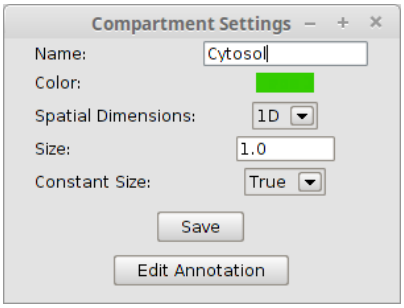


**Abbildung 3.19:** Die Abbildung zeigt das Menü zum Anlegen eines Zellkompartiments. Neben dem Namen des Zellkompartiments kann diesem auch eine Farbe zugeordnet werden. Diese kann genutzt werden, um alle Elemente in der Farbe ihres entsprechenden Zellkompartiments einzufärben. Die Eigenschaft *Spatial Dimension* legt die Anzahl der räumlichen Dimensionen des Zellkompartiments, die Option *Size* die Größe dieses fest. Über die Option *Constant Size* kann festgelegt werden, ob sich diese Größe ändern kann oder ob sie konstant ist.

### 3.4.3 Der Karteireiter *Analysis*

Der Karteireiter *Analysis* ermöglicht es, die berechneten T-Invarianten, P-Invarianten, MCT-Sets sowie MCS zu visualisieren. Im linken Teil der Abbildung 3.20 wird die grafische Oberfläche dieses Karteireiters, im rechten eine visualisierte T-Invariante gezeigt. Für jede Analysemethode gibt es jeweils einen gesonderten Bereich, welcher die verfügbaren Resultate bereitstellt.

Für die T-Invarianten findet eine weitere Unterteilung der Resultate statt. Neben der Auswahl aus allen T-Invarianten werden diese in verschiedene Kategorien unterteilt. Dabei wird zwischen trivialen T-Invarianten, solchen die, eine Input-Transition mit einer Output-Transition verbinden ($I/O$), an einer Input-Transition starten oder an einer Output-Transition enden, und zyklischen T-Invarianten unterschieden. Wird in einer dieser Listen eine T-Invariante ausgewählt, so werden alle darin enthaltenen Transitionen, deren Vor- und Nachplätze sowie alle Kanten zwischen diesen im *NetViewer* eingefärbt. Ein Beispiel hierfür ist im rechten Teil der Abbildung zu sehen. Die Standardfarbe, mit der diese Einfärbung vorgenommen wird, kann über die Optionen des *NetViewer* festgelegt werden und dort für jede der Analysemethoden, die hier bereitstehen, gesondert festgelegt werden. Über die Schaltfläche unter den Auswahllisten der T-Invarianten können diese als *Heatmap* visualisiert werden. Der Grad der Färbung hängt hierbei vom Faktor der einzelnen Transitionen in den T-Invarianten ab. Wird im Bereich der P-Invarianten eine solche ausgewählt, werden nur die daran beteiligten Plätze eingefärbt. Gleiches gilt für die MCT-Sets und die MCS. Hier werden nur die daran beteiligten Transitionen eingefärbt. Im unteren Bereich des Karteireiters befinden sich zwei Kontrollboxen. Wird die Kontrollbox *Highlight multiple selections* aktiviert, wird bei der Auswahl einer weiteren T-Invariante oder eines anderen Resultates die vorherige Einfärbung nicht verworfen. So ist es möglich

mehrere Resultate gleichzeitig einzufärben. Wird die Kontrollbox *Choose a new color for every selection* aktiviert, so kann bei jeder Auswahl eines Resultates die Farbe gewählt werden, mit der die Einfärbung stattfinden soll.

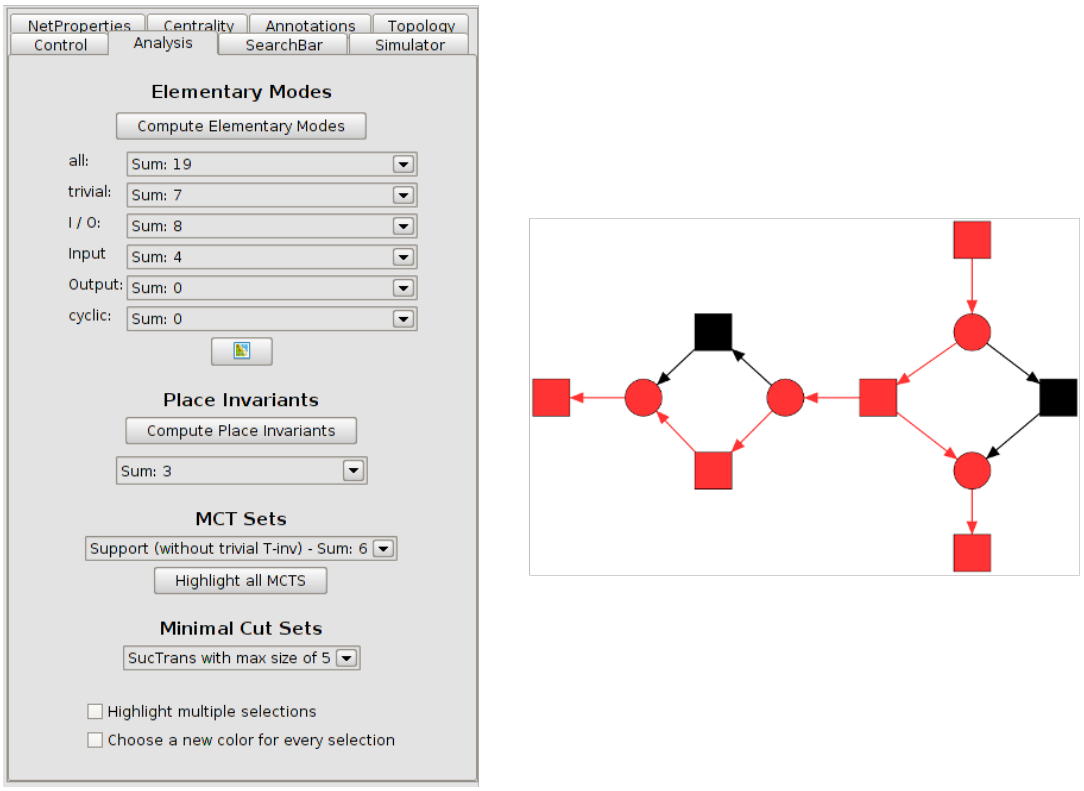


**Abbildung 3.20:** Die Abbildung zeigt den Karteireiter *Analysis* (links) und das Beispiel einer eingefärbten T-Invariante (rechts). Der Karteireiter ist in mehrere Teile unterteilt, welche jeweils die Resultate einer Analysemethode bereitstellen. Für die T-Invarianten findet eine Kategorisierung zwischen trivialen T-Invarianten, solchen die, eine Input-Transition mit einer Output-Transition verbinden (*I/O*), an einer Input-Transition starten oder an einer Output-Transition enden, und zyklischen T-Invarianten statt. Wird eine T-Invariante ausgewählt, wie im rechten Teil der Abbildung gezeigt, werden alle daran beteiligten Transitionen, deren Vor- und Nachplätze und die Kanten zwischen diesen eingefärbt. Bei der Auswahl eines Resultates der anderen Analysemethoden werden jeweils nur die enthaltenen Elemente eingefärbt. Über die Schaltfläche unter den Auswahllisten der T-Invarianten können diese als *Heatmap* visualisiert werden. Der Grad der Färbung hängt hierbei vom Faktor der einzelnen Transitionen in der T-Invariante ab. Für T-Invarianten und P-Invarianten gibt es zudem eine Schaltfläche, um diese direkt aus dem Karteireiter heraus zu berechnen. Die Schaltfläche *Highlight all MCTS* färbt alle vorhandenen MCT-Sets auf einmal ein. Die beiden Kontrollboxen im unteren Teil des Karteireiters dienen zur Steuerung des Einfärbeverhaltens.

Die Funktionen dieses Karteireiters stellen eine zentrale Motivation von MonaLisa dar. Neben der reinen Durchführung der Analysemethoden ist dies Motivation das Visualisieren deren Ergebnissen, möglichst direkt im PN. Der Karteireiter *Analysis* ermöglicht genau dies für zahlreiche Analysemethoden. Die Visualisierung der Ergebnisse, zum Beispiel der Invarianten oder MCT-Sets, helfen dem Modellierer bei der biologischen Interpretation dieser. Es müssen keine Textausgaben mehr betrachtet und manuell auf das

PN übertragen werden. Alle beteiligten Transitionen oder Plätze werden direkt im PN hervorgehoben und können anschließend auf biologische Funktionalität überprüft werden. Falls im Zuge einer Weiterentwicklung von MonaLisa weitere Analysemethoden implementiert werden sollten, so sollte das Prinzip, dass dessen Resultate visualisiert werden können, stets beachtet werden.

### 3.4.4 Der Karteireiter *SearchBar*

Der Karteireiter *SearchBar* beinhaltet jeweils eine Liste aller Plätze und aller Transitionen. Diese Listen können nach einer bestimmten Zeichenfolge gefiltert werden, sodass die Suche nach einzelnen Elementen möglich ist. In Abbildung 3.21 wird eine Übersicht über die grafische Oberfläche der *SearchBar* gegeben. Beide Listen sind alphabetisch aufsteigend sortiert. Wurden für einen Platz logische Plätze erzeugt, so ist dies in der Liste hinter dem entsprechenden Eintrag vermerkt. In beiden Listen ist per Rechtsklick ein Kontextmenü verfügbar, über welches zum Beispiel das PN auf diesen Platz zentriert werden kann, ein Zusammenlegen von Plätzen oder Transitionen möglich ist oder die einzelnen logischen Plätze angezeigt werden können. Wird in dem Textfeld im unteren Bereich eine Zeichenfolge eingegeben, so werden in den beiden Listen nur noch die Elemente angezeigt, in deren Namen die eingegebene Zeichenfolge enthalten ist.

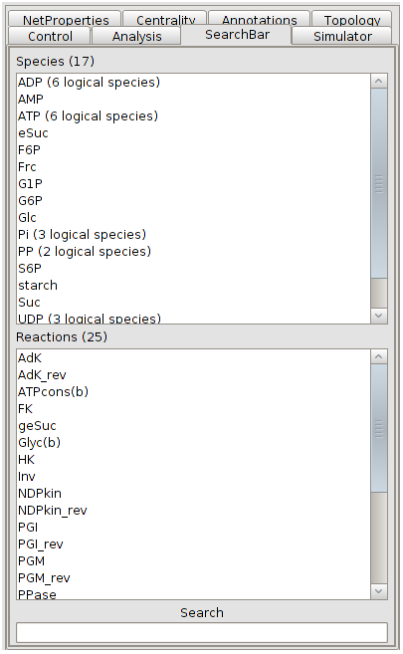


**Abbildung 3.21:** Die Abbildung zeigt den Karteireiter *SearchBar*. Die beiden Listen im oberen Teil enthalten jeweils alle vorhandenen Plätze oder Transitionen. Wird in diesen ein Rechtsklick auf ein Element ausgeführt, ist ein Kontextmenü verfügbar, über welches zum Beispiel das Zusammenführern mehrerer Plätze oder Transitionen verfügbar ist. Das Textfeld im unteren Bereich dient zum Filtern der beiden Listen. Wird hier eine Zeichenfolge eingegeben, so werden alle Elemente aus den Listen gelöscht, in deren Namen diese nicht enthalten sind.

### 3.4.5 Der Karteireiter *Simulator*

Der Karteireiter *Simulation* ermöglicht die Untersuchung des dynamischen Verhaltens des PN mit verschiedenen Methoden und basiert auf der abstrakten Klasse *AddonPanel*. Diese Erweiterung ist in Zusammenarbeit mit Pavel Balazki im Rahmen einer Bachelorarbeit und einer späteren Masterarbeit und Jörg Ackermann entstanden (Balazki et al., 2015). Im Folgenden werden einige Funktionen und Optionen dieser Erweiterung beschrieben. Eine ausführliche und vollständige Dokumentation dieser ist im Internet verfügbar (Simulator-Dokumentation, 2016). Abbildung 3.22 gibt eine Übersicht über die grafische Oberfläche des Karteireiters *Simulation*. Im oberen grün umrandeten Bereich befindet sich eine Auswahlliste, mit der zwischen den vier zur Verfügung stehenden Simulationsmodi gewählt werden kann. Zur Auswahl steht ein asynchroner und ein synchroner Modus sowie zwei stochastische Simulationsmodi. Alle vier Varianten werden in folgenden Abschnitten näher beschrieben. Unter dieser Auswahlliste befinden sich, analog zum Karteireiter *Control*, Menüelemente, mit denen Icon- oder Schriftgröße angepasst werden können sowie eine Schaltfläche zum Speichern der aktuellen Ansicht des PN als Bild. Diese Menüelemente sind erforderlich, da der Karteireiter *Control* während einer aktiven Simulation deaktiviert ist. So wird verhindert, dass während dieser keine Änderungen an der Topologie des PN vorgenommen werden kann.

Die blaue Umrandung markiert einen Bereich, in dem sich das für jeden Simulationsmodus spezifische Menü befindet. In der Abbildung wird das Menü des stochastischen Simulationsmodus *Mass Action Stochastic Simulation* gezeigt. Zum einen kann hier eine Endlos-Simulation gestartet werden, der *Continous mode*, welcher das PN simuliert, bis keine Transition mehr aktiviert ist, oder eine festgelegte Anzahl von Feuerungen simuliert werden. Zum anderen kann über die Schaltfläche *Enter simulation data* die Anzahl der Marken auf den Plätzen und die Feuerraten der Transitionen festgelegt werden.

Der rot umrandete Bereich beinhaltet allgemeine Funktionen der Erweiterung, die unabhängig vom Simulationsmodus sind. Über die Schaltfläche *Preferences* öffnet sich ein Menü zum Festlegen der allgemeinen Optionen der Erweiterung (siehe Abbildung 3.23). Dazu zählt unter anderem der Speicherort der Logdateien, wie oft die Ansicht im *NetViewer* bei einer längeren Simulation aktualisiert wird oder das Neusetzen des Anfangswertes des Zufallssimulators. Am Speicherort der Logdateien wird eine CSV-Datei angelegt, in der während einer Simulation in bestimmten Intervallen die aktuelle Markierung des PN gespeichert wird. So kann der Verlauf der Simulation nachvollzogen werden und auch mit externen Programmen ausgewertet werden. Die Änderung der Anzahl der Marken auf einer Auswahl von Plätzen über einen Simulationszeitraum kann aber auch direkt über die Schaltfläche *Show Plot* angezeigt werden. Hierzu wird die Java-Bibliothek *JFreeChart* verwendet (JFreeChart, 2016). Die Schaltfläche *Statistic* gibt eine Übersicht darüber, wie oft jede einzelne Transition bisher gefeuert hat. Eine aktuelle Markierung des PN kann über die Schaltfläche *Save marking* gespeichert werden. Über die Auswahlliste darüber kann anschließend zwischen den gespeicherten Markierungen gewechselt werden.

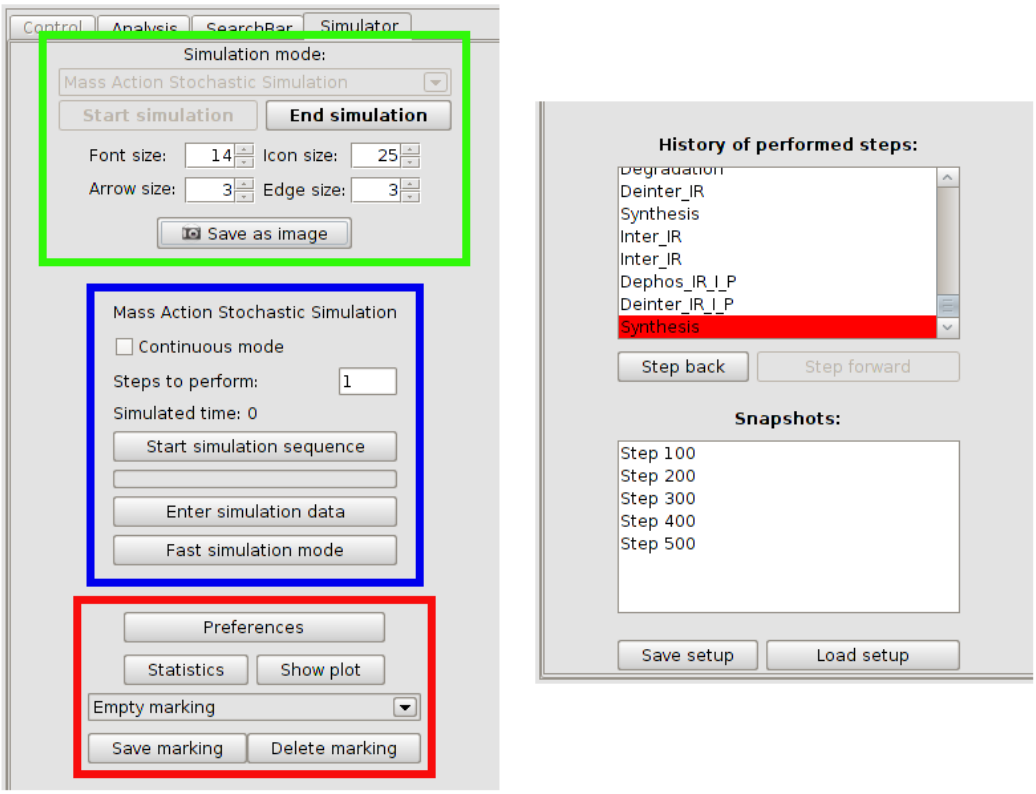


**Abbildung 3.22:** Die Abbildung zeigt die grafische Oberfläche des Karteireiters *Simulation*. Der grün umrandete Bereich beinhaltet zum einem die Auswahlliste der vier verschiedenen Simulationsmodi und zum anderen Elemente zum Anpassen zum Beispiel der Icon- und Schriftgröße. Eine Schaltfläche zum Speichern der aktuellen Ansicht im *NetViewer* in ein Bild (*Save as image*) ist hier auch enthalten. Im blau umrandeten Bereich befindet sich das simulationsspezifische Menü. Hier wird das Menü des *Mass Action Stochastic Simulation* Modus gezeigt. Über dieses ist sowohl das Starten einer Simulationssequenz möglich als auch eine Endlos-Simulation (*Continous mode*). Die Schaltfläche *Enter simulation data* ermöglicht das Festlegen der Feuerraten und der Anzahl der Marken auf den Plätzen. Allgemeine Einstellungen der Erweiterung werden im rot umrandeten Bereich festgelegt. Über die Schaltfläche *Statistic* ist eine aktuellle Statistik verfügbar, in der die Markierung des PN und für jede Transition die Anzahl ihrer Feuerungen aufgelistet werden. Ein Plot über den Simulationszeitraum und die Anzahl der Marken auf den Plätzen ist über *Show plot* abrufbar. Die Liste unter *History of performed steps* beinhaltet die letzten 100 gefeuerten Transitionen. In dieser Liste kann zu den jeweiligen Markierungen des PN zurückgewechselt werden. Nach jeweils 100 Simulationsschritten wird diese Liste geleert und in einer Momentaufnahme zusammengefasst. Diese wird dann in die Liste unter *Snapshots* hinzugefügt. Die Einstellungen einer Simulation, also Feuerraten und Markierung, können als XML Datei exportiert und zu einem späteren Zeitpunkt wieder geladen werden.

Im Bereich *Histroy of performed steps* werden jeweils die letzten 100 durchgeführten Simulationsschritte aufgeführt. Über die Schaltflächen *Step back* und *Step forward* oder durch das Auswählen eines Eintrages kann zwischen diesen Schritten gewechselt werden. Dabei wird die Markierung des PN auf den Zustand nach diesem Simulationsschritt zurückgesetzt. Der Bereich *Snapshot* beinhaltet eine Liste aller Momentaufnahmen. Eine Momentaufnahme wird nach jeweils 100 durchgeführten Simulationsschritten angelegt

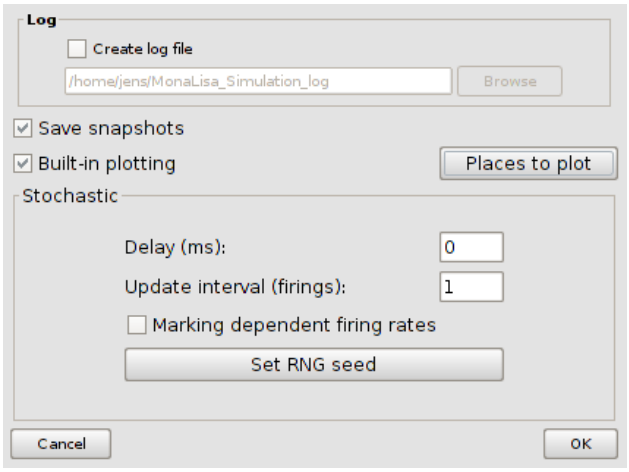


**Abbildung 3.23:** Die Abbildung zeigt die Optionen der Erweiterung *Simulation*. Der Ort, an dem die Logdateien der Simulationen abgelegt werden, kann im oberen Teil festgelegt werden. Die Kontrollbox *Save snapshots* steuert, ob für die letzten 100 Simulationsschritte eine Momentaufnahme angelegt wird. Über die Schaltfläche *Built-in plotting* kann die Funktion zum Plotten des Simulationsverlaufes aktiviert werden. Hierbei angezeigte Plätze können über die Schaltfläche *Places to plot* festgelegt werden. Im unteren Teil wird das Updaten des *NetViewer* bei längeren Simulationen gesteuert. Dieses kann entweder nach einer bestimmten Zeit oder nach einer Anzahl von Feuerungen geschehen. Ist die Kontrollbox *Marking dependent firing rates* aktiviert, so wird bei einer stochastischen Simulation die Wartezeit einer Transition auch von der Anzhal der Marken auf ihren Vorplätzen bestimmt. Die Schaltfläche *Set RNG seed* erzeugt einen neuen Anfangswert für den Zufallsgenerator.

und beinhaltet genau diese. Durch das Auswählen einer Momentaufnahme werden deren Simulationsschritte im oben beschriebenen Bereich *Histroy of performed steps* angezeigt, um ein Nachvollziehen dieser Schritte zu ermöglichen. Über die Schaltflächen *Save Setup* und *Load Setup* unten im Menü können die Einstellungen der Simulationsmodi in eine XML-Datei gespeichert und geladen werden. Dies ermöglicht es, verschiedene Szenarien zu simulieren, ohne jedes Mal die Parameter manuell anpassen zu müssen.

Die Simulation des PN wird von dieser Erweiterung nicht nur berechnet sondern auch im *NetViewer* visualisiert. Dazu zählt das Anzeigen der vorhandenen Marken auf den Plätzen und das Hervorheben aktivierter Transitionen. Ein Beispiel hierfür ist in Abbildung 3.24 zu sehen. Die auf einem Platz vorhandenen Marken werden durch rote Punkte repräsentiert oder bei mehr als vier Marken durch die entsprechende Zahl. Aktivierte Transitionen werden mit einer grünen Umrandung hervorgehoben. Die zuletzt gefeuerte Transition ist durch eine rote Umrandung markiert. Ist diese noch immer aktiviert, so ist sie grün ausgefüllt. Neben der automatisierten Simulation über eine bestimmte Anzahl von Schritten oder einem bestimmten Zeitraum können aktivierte Transitionen auch manuell im *NetViewer* durch anklicken gefeuert werden.

Mit dem Karteireiter *Simulation* stellt MonaLisa auch eine nicht strukturelle Analysemethode zu Verfügung. Der Verlauf und das Ergebnis der Simulation des PN hängt nicht nur von dessen Topologie ab, sondern auch von der Markierung des PN zu Beginn dieser. Die Anwendbarkeit solcher Simulationen für metabolische Modelle wurde von H. Genrich

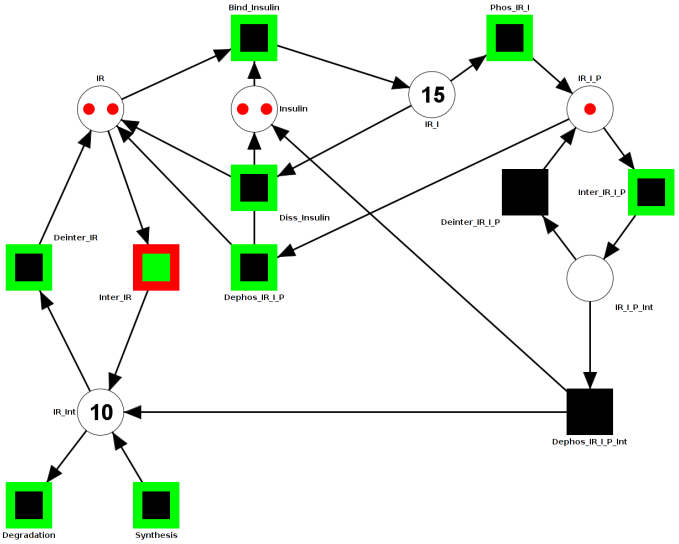


**Abbildung 3.24:** Die Abbildung zeigt die Visualisierung der Simulation eines PN. Die Anzahl der Marken auf einem Platz werden durch rote Punkte oder bei mehr als vier Marken durch die entsprechende Zahl repräsentiert. Aktivierte Transitionen werden mit einer grünen Umrandung markiert. Eine rote Umrandung kennzeichnet die zuletzt gefeuerte Transition. Ist diese grün ausgefüllt, ist sie noch immer aktiv.

(2001) gezeigt.

Neben der Simulation des dynamischen Verhaltens eines PN ist auch dessen Erreichbarkeitsgraph (Mayr, 1984) von Interesse. In diesem werden alle Markierungen als Knoten notiert, die von einer gegebenen Startmarkierung aus erreicht werden können. Eine Kante zwischen zwei Knoten zeigt an, dass durch das Feuern einer Transition das PN von einer Markierung in die andere überführt werden kann. Das Berechnen des Erreichbarkeitsgraphen ist *EXPSPACE-schwer*, sowie *NP-schwer* (Esparza, 1998), und somit keine triviale Angelegenheit. Grundlegende Arbeiten zur Implementierung des Erreichbarkeitsgraphen in MonaLisa fanden im Rahmen einer Bachelorarbeit statt (Hülsdunk, 2013). Das Fertigstellen dieser Implementation wäre ein nächster Schritt für die künftige Weiterentwicklung von MonaLisa. Mit Hilfe des Erreichbarkeitsgraphen könnte dann die *Lebendigkeit* (Starke, 1990) eines PN gezeigt werden. Diese Eigenschaft ist ein wichtiges Merkmal bei der Validierung eines biologischen PN (Heiner et al., 2004; Koch und Heiner, 2008). Ergänzt werden könnte diese Analysemethode um die stark modifizierte Variante zur Berechnung des Erreichbarkeitsgraphen für eine spezielle Form von PN, vorgestellt in Nöthen (2014).

### Zufallszahlengenerator

Für beide stochastischen Simulationsmodi ist ein Zufallszahlengenerator notwendig. Hierzu wird in diesem Fall nicht auf die in Java vorimplementierten Lösungen zurückgegriffen. Der hier verwendete Zufallszahlengenerator ist eine Java-Implementierung eines in Press et al. (2007) vorgeschlagenen Vorgehens. Es ist ein zusammengesetzter Generator, welcher aus zwei XORShift-Generatoren besteht, die mit einem Kongruenzgenerator und einem

Paralleladdierer mit Übertragsvorausberechnung kombiniert werden (Coffey, 2016).

### Konstante Plätze und mathematische Ausdrücke

Beide stochastischen Simulationsmodi lassen die Definition von konstanten Plätzen zu. Die Anzahl der Marken auf diesen Plätzen wird durch das Feuern einer Transitionen nicht verändert. Die Anzahl der Marken auf solchen Plätzen kann entweder fest definiert werden oder aber durch mathematische Ausdrücke in Abhängigkeit der Zeit oder der Anzahl von Marken auf anderen Plätzen gemacht werden. Sie helfen zum Beispiel externe Faktoren, inhibitorische Effekte oder Grenzbedingungen zu modellieren. Ein Beispiel für einen solchen mathematischen Ausdruck ist

```
if Time <= (4 * 60) then 2000;
if Time <= (15 * 60) then 500.
```

Hierbei ist die Anzahl der Marken abhängig von der Zeit. In den ersten 4 Minuten der Simulation wird die Anzahl von Marken auf dem Platz auf 2000 festgelegt. Danach wird sie auf 500 reduziert und nach 15 Minuten auf 0. In Tabelle 3.2 sind alle unterstützten mathematischen Ausdrücke aufgeführt. Diese reichen von einfachen Rechenoperatoren, über Auf- und Abrunden bis zu den Winkelfunktionen. Die Auswertung der mathematischen Ausdrücke erfolgt mit Hilfe der Java-Bibliothek *exp4j* (exp4j, 2016).

### Der *Asynchronus*-Modus

Im *Asynchronus*-Modus wird pro Simulationsschritt nur eine einzige, zufällig bestimmte (aktivierte) Transition gefeuert, wobei alle aktivierten Transitionen die gleiche Feuerwahrscheinlichkeit besitzen.

Diese Methode stellt die einfachste Methode dar, um ein PN zu simulieren. Es existieren jedoch keinerlei Parameter, um die Simulation zu steuern oder zu beeinflussen. Mit dieser Methode kann jedoch ein erstes Überprüfen der Schaltlogik des PN stattfinden.

### Der *Synchronus*-Modus

Beim *Synchronus*-Modus wird pro Simulationsschritt nicht nur eine aktivierte Transition gefeuert, sondern der Algorithmus versucht alle aktivierten Transitionen zu feuern. Dies ist nicht immer möglich, da aktivierte Transitionen, welche sich einen Vorplatz teilen, um dessen Marken konkurrieren. Das Feuern einer dieser Transitionen kann nun zur Inaktivierung der anderen Transitionen führen. In einem solchen Fall wird so lange jeweils zufällig eine dieser Transitionen gefeuert, bis alle Marken auf dem Platz verbraucht wurden oder keine dieser Transitionen mehr aktiv ist. Die Feuerwahrscheinlichkeit jeder Transition ist dabei identisch. Der Modus erlaubt zudem, dass nur ein bestimmter, vorher festgelegter, Prozentsatz aller aktivierten Transitionen pro Simulationsschritt gefeuert wird.

| Operation oder Funktion | Beschreibung |
|---|---|
| Addition | $2 + 2$ |
| Subtraktion | $2 - 2$ |
| Multiplikation | $2 \cdot 2$ |
| Division | $2/2$ |
| Potenzieren | 2 ^ 2 |
| Vorzeichen Operatoren | $+2 - (-2)$ |
| Modulo | 2 % 2 |
| abs | absoluter Wert |
| acos | Arkuskosinus |
| asin | Arkussinus |
| atan | Arkustangens |
| cbrt | Kubikwurzel |
| ceil | aufrunden |
| cos | Kosinus |
| cosh | hyperbolischer Kosinus |
| exp | $e^x$ |
| floor | abrunden |
| log | natürlicher Logarithmus |
| sin | Sinus |
| sinh | hyperbolischer Sinus |
| sqrt | Quadratwurzel |
| tan | Tangens |
| tanh | hyperbolischer Tangens |
| div(x,y) | Integer Division |

**Tabelle 3.2:** Operatoren und Funktionen, welche zur Erstellung von mathematischen Ausdrücken in den beiden stochastischen Simulationsmodi von MonaLisa zur Verfügung stehen. (Verändert nach Balazki et al. (2015))

Diese Methode führt einen Parameter in die Simulation ein. Jedoch betrifft dieser nicht die einzelnen Transitionen, sondern ist ein globaler Parameter. Die Parametrisierung einzelner Transitionen und Plätze wird durch die folgenden beiden Methoden implementiert.

### Der *Stochastic Simulation*-Modus

Der *Stochastic Simulation*-Modus ist einer der beiden stochastischen Simulationsmodi. Ist in diesem Modus eine Transition aktiviert, so muss diese eine definierte Zeit $dt$ warten, bevor sie feuern darf. Diese Wartezeit $dt$ wird für jede Transition einzeln als exponentiell verteilte Zufallsvariable $Exp(\lambda)$ simuliert. Die Feuerrate $\lambda$ einer Transition kann frei festgelegt werden. Zusätzlich ist es möglich, diese Wartezeit in Abhängigkeit zu der Anzahl von vorhandenen Marken auf den Vorplätzen einer Transition zu setzen. Eine detaillierte Beschreibung dieses Vorgehens ist in Wilkinson (2011) zu finden. Die Simulation ist dadurch abhängig von der Zeit, weshalb während dieser ein Zähler die vergangene Zeit mitzählt. Für einen Simulationsschritt wird aus allen aktivierten Transitionen diejenige gefeuert,

welche die niedrigste, verbleibende Wartezeit hat. Trifft dies auf mehrere Transitionen zu, so wird aus diesen zufällig eine Transition bestimmt. Nach dem Feuern einer Transition wird die Simulationszeit um die Wartezeit der Transition erhöht und die Wartezeiten aller aktivierten Transitionen neu bestimmt. Die Simulation wird dadurch beschleunigt, dass die Wartezeit nur für die Transitionen neu berechnet wird, auf deren Vorplätzen sich die Anzahl der Marken verändert hat.

Durch die Parametrisierung der einzelnen Transitionen ist nun eine genauere Simulation des biologischen Systems möglich. Der Fortschritt der Simulation wird zudem in Zeiteinheiten gemessen, nicht mehr in Simulationsschritten, die zeitlos sind. Die Methode lässt komplexere Simulationen zu, beachtet jedoch nicht, wie viel der einzelnen Substanzen vorhanden sind. Um diesen Faktor mit in die Simulation einzubeziehen, wurde die folgende Methode implementiert, die zu diesem Zweck eine Parametrisierung der Plätze zulässt.

#### Der *Mass Action Stochastic Simulation*-Modus

Als letzter Simulationsmodus wurde der *Mass Action Stochastic Simulation*-Modus implementiert. Dieser ermöglicht eine Simulation durch den exakten und approximativen *Stochastic Simualtion Algorithm* (SSA). Zur Berechnung der Feuerzeit $\tau_1$ des approximativen Algorithmus wird die Formel 24 des Kapitels 3.2 aus Cao (2010) verwendet. Die Anzahl der Marken auf den Plätzen kann entweder direkt gesetzt werden oder als Konzentration angegeben werden. In letzterem Fall bedarf es der Definition eines Reaktionsvolumens des Systems, um aus den Konzentrationen die Anzahl der Marken zu berechnen.

Eine weitere Funktion des *Mass Action Stochastic Simulation*-Modus ist der *Fast simualtion mode*. Dieser dient zur parallelen Simulation eines bestimmten Zeitraumes. Eine Übersicht über den *Fast simulation mode* wird in Abbildung 3.25 gegeben. Zu sehen ist ein Karteireiter dieses Modus. Jeder Karteireiter steht für ein unabhängiges Experiment, deren Parameter unabhängig voneinander sind. Für ein Experiment muss zuerst der zu simulierende Zeitraum (*Time span to simulate*) festgelegt werden und auch in welchen Abständen in der Logdatei die aktuelle Markierung gespeichert werden soll (*Output Interval*). Über die Auswahlliste *Select algorithm* kann zwischen dem exakten und dem approximativen SSA gewählt werden. Über den Menüpunkt *Number of parallel simulations* kann festgelegt werden, wie viele Simulationen von diesem Experiment gleichzeitig durchgeführt werden sollen. Die Schaltfläche *Refresh seed* setzt einen neuen Anfangswert für den Zufallsgenerator. Im unteren Teil befindet sich ein Informationsbereich für jede der parallelen Simulationen. Hier wird zum Beispiel der aktuelle Fortschritt der Simulation angezeigt. Über die Schaltflächen neben dieser Anzeige lässt sich zum Beispiel die Endmarkierung der Simulation auf das PN übertragen (*Apply to PN*) oder der Simulationsverlauf als Plot anzeigen (*Show plot*).

Diese Methode ermöglicht es Plätze und Transitionen mit Parametern zu versehen, die Einfluss auf die Simulation haben. Gillespie's Methode berücksichtigt, dass für das Stattfinden einer Reaktion auch das Zusammentreffen der Reaktanten stattfinden muss.

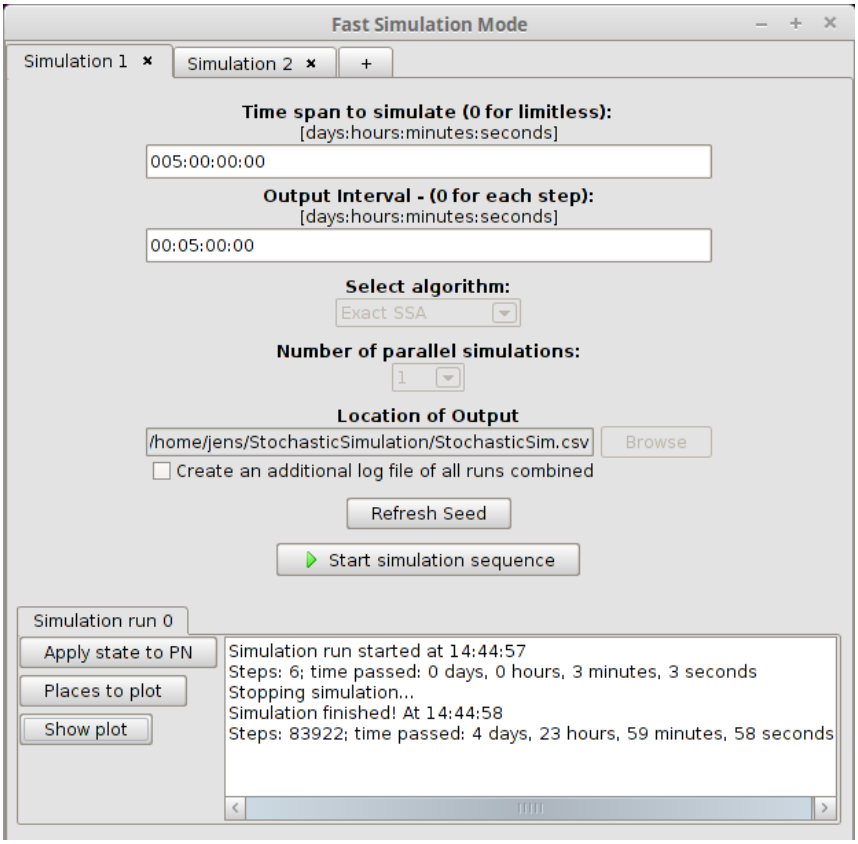


**Abbildung 3.25:** Die Abbildung gibt eine Übersicht über das Menü des *Fast Simulation Mode*. Jeder Karteireiter in diesem Menü repräsentiert ein unabhängiges Experiment. Für jedes Experiment kann zunächst die zu simulierende Zeit (*Time span to simulate*) festgelegt werden und in welchem Abstand die aktuelle Markeriung in die Logdatei übertragen wird (*Output Interval*). Mit der Auswahlliste *Select algorithm* kann zwischen dem exakten und approximativen SSA gewählt werden. Darunter lässt sich die Anzahl der parallelen Simulationen für dieses Experiment bestimmen. Mit der Schaltfläche *Refresh seed* kann ein neuer Anfangswert für den Zufallsgenerator erzeugt werden. Im unteren Bereich befindet sich die Ausgabe jeder einzelnen parallelen Simulation. Mit den hier vorhandenen Schaltflächen kann die Endmarkierung der Simulation auf das PN übertragen werden (*Apply to PN*) oder der Verlauf der Simulation als Plot angezeigt werden (*Show plot*).

Von den in MonaLisa implementierten Simulationsmethoden abstrahiert diese Methode also die realen Vorgänge am genausten. Dieser Vorteil der Methode ist auch ein Nachteil, denn damit die Simulation durchgeführt werden kann, müssen experimentelle Informationen über Stoffkonzentration und Reaktionsgeschwindigkeit vorliegen. Liegen diese nicht vor, müssen diese Werte geschätzt werden, was zwar zu einem Ergebnis führt, dieses aber erst experimentell bestätigt werden muss. Eine Anwendung dieser Methode ist in Balazki et al. (2015) zu finden, in welcher die Autoren ein Modell des Recycling-Prozesses des Insulinrezeptors im Menschen unter verschiedenen Bedingungen untersuchen.

Durch das Bereitstellen des *Fast Simulation Mode* ist es leicht möglich, verschiedene Ausgangsszenarien des PN zu simulieren. Da es sich um eine stochastische Simulation handelt, ist es wichtig ein und dasselbe Szenario öfter zu wiederholen, um einen möglichst großen Bereich des Ergebnisraums abzudecken. Auch dies ermöglicht der *Fast Simulation*

*Mode*. Zur Zeit macht der *Fast Simulation Mode* nur die Auswertung jeder Simulation einzeln möglich. Hier wäre die Überlegung, eine Möglichkeit zu schaffen, den Durchschnitt aller Simulationen zu ermitteln und nur diesen als Plot bereitzustellen.

### 3.4.6 Der Karteireiter *Topology*

Der Karteireiter *Topology* ermöglicht die Untersuchung der Verteilung der Knotengrade des PN und basiert auf der abstrakten Klasse *AddonPanel*. Eine Übersicht über das Menü dieser Erweiterung ist im linken Teil der Abbildung 3.26 zu sehen. Im oberen Teil befindet sich die tabellarische Auflistung aller Plätze und Transitionen mit ihren entsprechenden Knotengraden. Diese Tabellen können durch das Auswählen einer Spalte nach den Werten dieser sortiert werden. Durch das Selektieren des Namens einer Transition oder eines Platzes wird dieser im *NetViewer* hervorgehoben. Neben der Auflistung des Knotengrades für jeden einzelnen Knoten ist auch die Verteilung des Knotengrades interessant. Hierbei wird auf der x-Achse der Knotengrad $k$ aufgetragen und auf der y-Achse die Wahrscheinlichkeit $p(k)$, dass ein Knoten diesen Knotengrad besitzt. Diese Verteilungen können über die

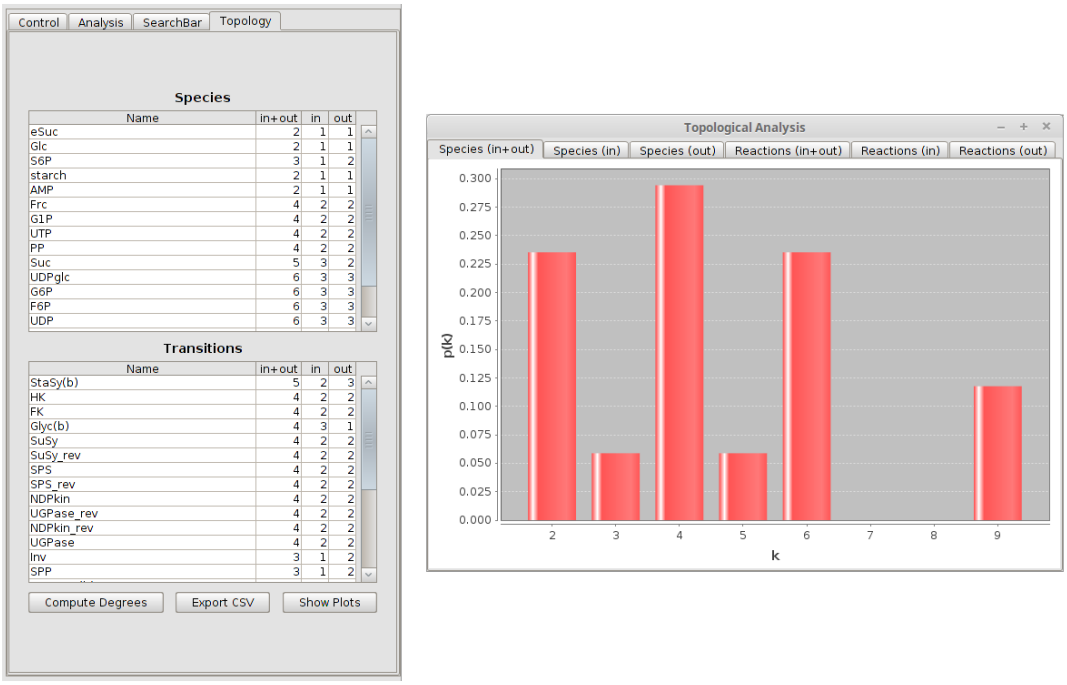


**Abbildung 3.26: Links:** Menü des Karteireiters *Topology*. Dieses enthält die beiden sortierbaren Tabellen mit den Knotengraden der Plätze (oben) und Transitionen (unten). Die Schaltfläche *Compute Degrees* füllt diese Tabellen mit den aktuellen Werten. Über die Schaltfläche *Export CSV* ist ein Export dieser Werte in eine CSV-Datei möglich. Die Verteilung der Knotengrade kann über die Schaltfläche *Show Plots* angezeigt werden.
**Rechts:** Ein Diagramm zur Verteilung der Knotengrade. Auf der x-Achse des Balkendiagramms ist der Knotengrad aufgetragen, auf der y-Achse die Wahrscheinlichkeit, mit der ein Knoten im PN diesen Knotengrad besitzt. Das PN, welches als Grundlage der hier gezeigten Werte dient, ist ein Modell der Verarbeitung von Saccharose in der Wurzel der Kartoffelpflanze (Koch et al., 2005).

Schaltfläche *Show Plots* in Form von Balkendiagrammen dargestellt werden. Ein Beispiel dazu wird im rechten Teil der Abbildung gezeigt. Zur Visualisierung der Diagramme wird die Bibliothek *JFreeChart* (JFreeChart, 2016) benutzt. Diese stellt umfangreiche Optionen zur Verfügung, die es dem Benutzer erlauben, zur Laufzeit die Diagramme anzupassen, wie zum Beispiel die Achsenbeschriftung oder Auswahl der Farben, diese zu drucken oder als Bild zu speichern. Neben der visuellen Darstellung der Verteilungen können diese über die Schaltfläche *Export CSV* auch in eine CSV-Datei exportiert werden.

### 3.4.7 Der Karteireiter *Centrality*

Der Karteireiter *Centrality* ermöglicht die Berechnung verschiedener Zentralitätsmaße und basiert auf der abstrakten Klasse *AddonPanel* und wurde in Zusammenarbeit mit Lilya Mirzoyan implementiert (Mirzoyan, 2013). Es wurden vier Zentralitätsmaße implementiert: die *Closeness-Zentralität*, die *Exzentrizität-Zentralität*, die *Betweenness-Zentralität* und die Eigenvektor-Zentralität.

Diese werden jeweils für Plätze und Transitionen getrennt berechnet, weshalb zwei getrennte Adjazenzmatrizen benötigt werden, die jeweils nur aus Plätzen oder Transitionen bestehen. In dieser Adjazenzmatrix ist jeweils vermerkt, welche Plätze oder Transitionen direkte Nachbarn sind. Die Plätze $p_i$ und $p_j$ sind direkte Nachbarn, falls eine Transition $t$ existiert, für die gilt $t \in {\bullet}p_i \cap p_j \bullet \ \vee \ p_i \bullet \cap \bullet \, p_j$. Analog sind die Transitionen $t_i$ und $t_j$ direkte Nachbarn, falls ein Platz $p$ existiert, für den gilt $p \in {\bullet}t_i \cap t_j \bullet \ \vee \ t_i \bullet \cap \bullet \, t_j$. Die Berechnung aller kürzesten Wege zwischen allen Knotenpaaren, welche für einige der Zentralitätsmaße benötigt werden, wird durch eine Implementierung des Floyd-Warshall-Algorithmus (Floyd, 1962) realisiert.

Die Abbildung 3.27 zeigt das Menü des Karteireiters *Centrality*. Im oberen Teil befinden sich die beiden Tabellen für das Ranking der Plätze und Transitionen. Eine Tabelle beinhaltet jeweils alle Plätze oder Transitionen des aktuellen PN und je eine Spalte für die vier Zentralitätsmaße. Die Tabellen können durch das Auswählen eines Zentralitätsmaßes nach diesem sortiert werden. Die Auswahl eines Platzes oder einer Transition hebt diesen im *NetViewer* hervor. Im unteren Teil des Menüs befinden sich die Schaltflächen zur Steuerung der Erweiterung. Die Schaltfläche *Calculate Ranking* startet die Berechnung der Zentralitätsmaße auf Grundlage des aktuellen PN und trägt die Resultate in die Tabellen ein. Über die Schaltfläche *Export Results* können die Resultate in CSV-Dateien exportiert werden. In diesem sind jeweils alle Knoten eines Typs sowie die Werte der einzelnen Zentralitätsmaße aufgelistet. Über die Schaltfläche *Heatmap for* wird die im benachbarten Auswahlmenü gewählte Zentralität als *Heatmap* auf die Plätze und Transitionen visualisiert.

Mit Hilfe der beiden Karteireitern *Topology* und *Centrality* können die topologischen Eigenschaften des PN untersucht werden. Streng genommen stellt auch der Knotengrad ein Zentralitätsmaß dar. Die Berechnung der Zentralitätsmaße macht ein Ranking der Plätze und Transitionen möglich. Die Aussage eines solchen Rankings hängt vom betrachteten

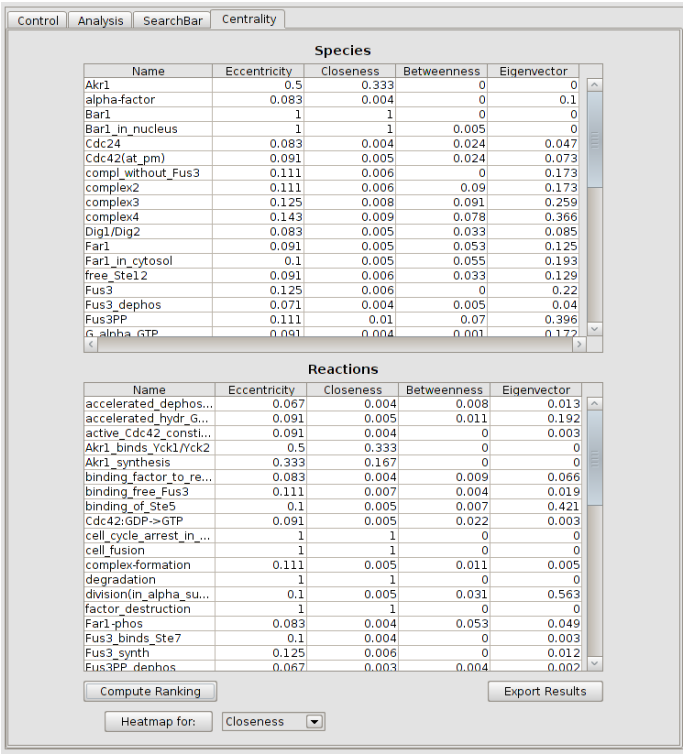


**Abbildung 3.27:** Die Abbildung zeigt das Menü des Karteireiters *Centrality*. Im oberen Teil befinden sich die beiden sortierbaren Tabellen aller Plätze (oben) und Transitionen (unten) und die dazugehörigen Werte der verschiedenen Zentralitätsmaße. Die Schaltfläche *Compute Ranking* dient dazu, diese Tabellen auf der Basis des aktuellen PN zu füllen. Über die Schaltfläche *Export Results* können diese Werte dann in eine CSV-Datei exportiert werden. Die Schaltfläche *Heatmap for:* dient im Zusammenhang mit dem nebenstehenden Auswahlmenü zur Visualisierung der ausgewählten Zentralität als *Heatmap* im *NetViewer*. Das PN, welches als Grundlage der hier gezeigten Werte dient, ist ein Modell des Paarungspheromon-Antwort-Signalweges in *Saccharomyces cerevisiae* (Sackmann et al., 2006).

Zentralitätsmaß ab. Weist ein hoher Knotengrad auf eine Beteiligung an vielen Reaktionen oder eine komplexe Reaktion hin, so deutet ein hoher Wert bei der *Betweenness-Zentralität* zum Beispiel auf eine Transition hin, die eine zentrale Rolle in der Kommunikation zwischen anderen Knoten spielt. Im Kontext eines Signaltransduktionsweges weist zum Beispiel ein hoher Wert in der *Closeness-Zentralität* auf ein zentrales Element hin, dessen Regulierung sich schnell auf viele weitere Elemente auswirkt. Eine solche Transition eignet sich also gut als Ansatzpunkt für die Regulierung des ganzen Signaltransduktionsweges. Durch die Bereitstellung dieser Analysemethoden wird es ermöglicht, die Bedeutung der Transitionen und Plätze auf spezielle Fragestellungen hin zu untersuchen und Schlüsselfiguren in PN zu identifizieren. Eine Anwendung der Analyse eines genregulatorischen Netzwerks mit Hilfe von Zentrailitätsmaßen wird von Koschützki und Schreiber (2008) gezeigt. Aber auch bei Protein-Protein-Interaktions Modellen findet die Betrachtung der Zentralitäten Anwendung (Wuchty und Stadler, 2003).

Neben der Berechnung der vorhandenen Zentralitätsmaße wäre für eine zukünftige

Entwicklung von MonaLisa die Ergänzung um weitere Zentralitätsmaße denkbar, um neue Fragestellungen an das PN stellen zu können. Anbieten würde sich hier zum Beispiel die Implementierung des Katz-Index (Katz, 1953).

### 3.4.8 Der Karteireiter *NetProperties*

Mit Hilfe des Karteireiters *NetProperties* lässt sich das PN auf elementare Netzeigenschaften überprüfen. Der Karteireiter basiert auf der abstrakten Klasse *AddonPanel* und wurde in Zusammenarbeit mit Daniel Noll entwickelt (Noll, 2013). Eine Übersicht über diesen Karteireiter gibt die Abbildung 3.28. Im oberen Abschnitt *One-sided nodes* können durch Aktivieren der Kontrollbox hinter den entsprechenden Einträgen Transitionen ohne Vor- oder Nachplätze sowie Plätze ohne Vor- oder Nachtransitionen farblich hervorgehoben werden. Die hierfür verwendete Farbe kann über den Bereich hinter den jeweiligen Kontrollboxen festgelegt werden. Im unterem Abschnitt lassen sich die elementaren Netzeigenschaften auswählen, auf die das PN überprüft werden soll. Die Überprüfung wird über die Schaltfläche *Check properties* gestartet. Nach Abschluss der Überprüfung wird

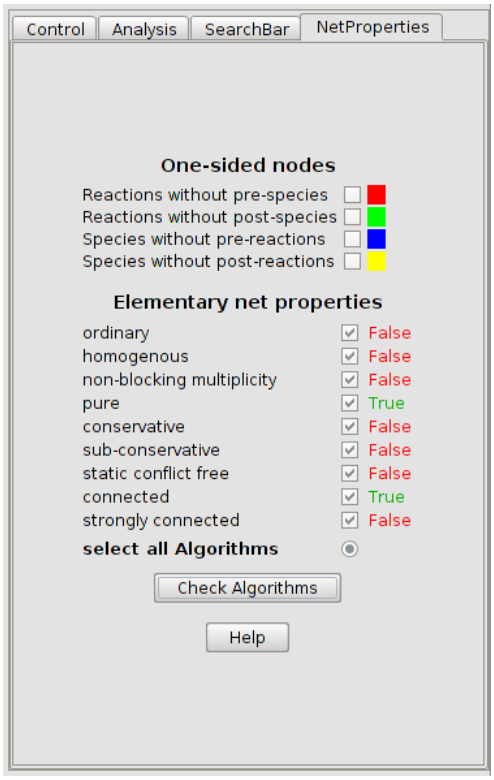


**Abbildung 3.28:** Die Abbildung zeigt die Oberfläche des Karteireiters *NetProperties*. Im oberen Teil, zusammengefasst im Block *One-sided nodes*, können diese im *NetViewer* farblich hervorgehoben werden, die Farbe hierzu kann über das Feld hinter den jeweiligen Einträgen festgelegt werden. Im unteren Teil, im Block *Elementary net properties*, können diejenigen Eigenschaften ausgewählt werden, welche durch die Schaltfläche *Check Algorithms* getestet werden. Nachdem die ausgewählten Netzeigenschaften getestet wurden, wird hinter den jeweiligen Eigenschaften angezeigt, ob diese erfüllt wird (*True*) oder nicht (*False*). Als Grundlage für diese Abbildung dient ein PN der Verarbeitung von Saccharose in der Wurzel der Kartoffelpflanze (Koch et al., 2005).

hinter dem jeweiligen Eintrag der Netzeigenschaften angezeigt, ob diese vom PN erfüllt wird (*True*) oder nicht (*False*). Über die Schaltfläche *Help* lässt sich ein neues Fenster öffnen, in welchem die Netzeigenschaften näher erläutert werden.

### 3.4.9 Der Karteireiter *Annotation*

Das PN kann über den Karteireiter *Annotation* mit MIRIAM-Identifikatoren und SBO-Begriffen annotiert werden. Dieser Karteireiter basiert auf der abstrakten Klasse *Addon-Panel*. Eine Übersicht über diesen Karteireiter gibt Abbildung 3.29. Die linke Seite der Abbildung zeigt des Menü zur Annotation des Modells selbst. Im oberen Teil ist das Benennen des Modells sowie das Hinzufügen von MIRIAM-Identifikatoren möglich. Dazu muss die Datenbank ausgewählt werden, zu welcher die Annotation führen soll, als auch der eindeutige Identifikator aus dieser angegeben werden. Zusammen mit dem Typ der Annotation bilden diese Angaben den vollständigen MIRIAM-Identifikator. Das SBML-Format bietet zudem die Möglichkeit, die Autoren, welche am Modell mitgearbeitet haben, zu hinterlegen, sowie eine *History* anzulegen, die angibt, zu welchen Zeitpunkten neue Versionen des Models veröffentlicht wurden. Diese Informationen können im mittleren und unteren Teil eingeben und bearbeitet werden.

Der rechte Teil der Abbildung zeigt das Menü zur Annotation von Plätzen und Transitionen. Wird im *NetViewer* ein Element ausgewählt, so ist hier das Bearbeiten seiner Annotation möglich. Diese erfolgt nach dem gleichen Schema wie bei der Annotierung des Modells selbst. Mit einem Rechtsklick auf einen Identifikator in der Liste ist es möglich, diesen zu bearbeiten oder zu löschen. Ergänzend zum MIRIAM-Identifikator ist hier das Zuordnen eines SBO-Begriffes möglich. Ein dazu analoges Menü dient zur Annotierung der Zellkompartimente, welches über den Karteireiter *Control* und der dortigen Liste aller Zellkompartimente verfügbar ist.

Das Auswahlmenü der Quelldatenbank für den MIRIAM-Identifikator beinhaltet alle offiziell unterstützten Datenbanken. Diese Informationen bezieht MonaLisa aus einer XML-Datei, in welcher all diese Datenbanken hinterlegt sind. Zu jeder Datenbank enthält die Datei zusätzlich einen regulären Ausdruck, um den angegebenen eindeutigen Identifikator zu verifizieren. Diese werden verwendet, um die Eingaben zu überprüfen, wodurch sicher gestellt wird, dass nur zulässige und gültige MIRIAM-Identifikatoren eingetragen werden können. Die hier hinterlegten Annotationen werden nur beim Exportieren des PN in das SBML-Format verwendet, da die andere von MonaLisa unterstützten Formate solche Informationen nicht unterstützen.

Wird mit MonaLisa ein PN aus einer SBML-Datei eingelesen, in der eine Annotation der Elemente hinterlegt ist, so wird diese Annotation während des Imports übernommen. Wird nun ein Element des PN ausgewählt, so werden diese Annotationen direkt angezeigt und können bearbeitet werden. Um einen schnellen Zugriff auf den MIRIAM-Identifikator zu erhalten, kann über einen Rechtsklick auf diesen der entsprechende Eintrag im Browser aufgerufen werden.

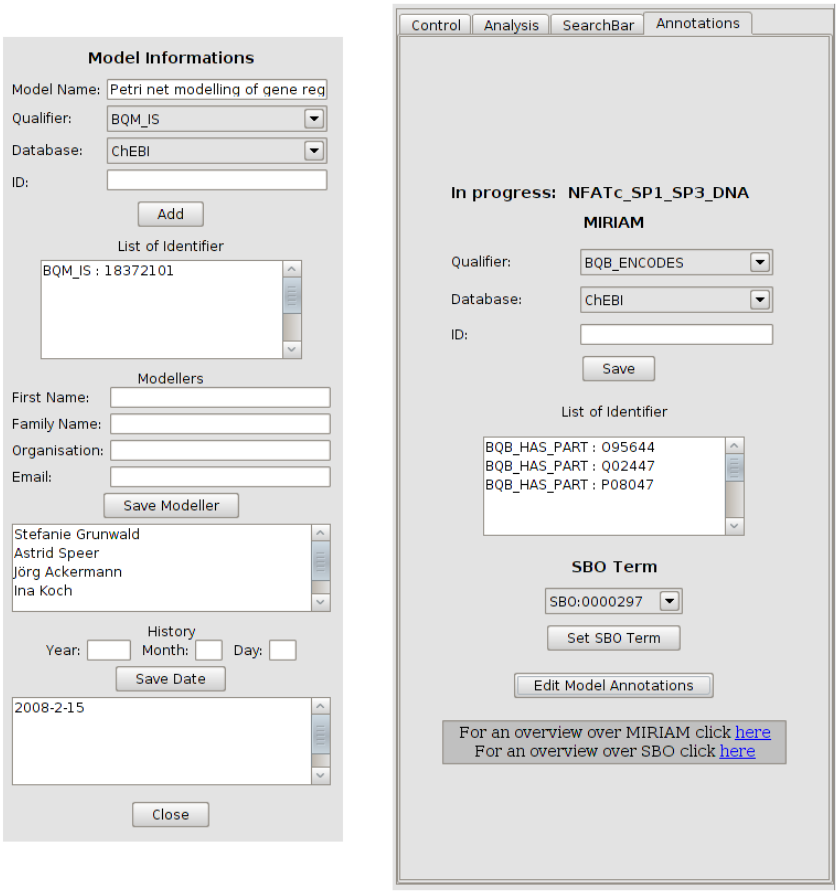


**Abbildung 3.29:** Die Abbildung zeigt die zwei verschiedenen Menüs des Karteireiters *Annotation*. **Links:** Das Menü zur Annotation des Modells. Neben dem Eintragen von MIRIAM-Identifikatoren (oberer Teil) ist auch das Hinterlegen der Personen möglich, die an diesem Modell mitgearbeitet haben (mittlerer Teil) sowie das Anlegen eines Verlaufes (unterer Teil).
**Rechts:** Menü zur Annotierung der Plätze und Transitionen des Modells. Wird im *NetViewer* ein Element ausgewählt, so ist hier die Bearbeitung seiner Annotaion möglich. Das Feld *In progress* zeigt hierbei an, welches Element gerade bearbeitet wird. Zusätzlich zu den MIRIAM-Identifikatoren können die einzelnen Elemente noch mit einem SBO-Begirff versehen werden (Bereich *SBO Term*). Im unteren Teil ist das Abrufen von Informationen über MIRIAM und SBO möglich. Grundlage für diese Abbildung ist ein PN-Modell der Genregulierung der Duchenne Muskeldystrophie (Grunwald et al., 2008).

Soll ein Modell veröffentlicht oder mit anderen Leuten geteilt werden, so ist es notwendig, dass die Elemente des PN eindeutig zu identifizieren sind. Bei der existierenden Vielzahl an bekannten Genen, Proteinen oder Metaboliten sind Abkürzungen nicht mehr eineindeutig und auch die Nomenklatur zur Namensgebung von Person zu Person unterschiedlich. Um eine eineindeutige Identifikation der Elemente eines PN zu ermöglichen, wurden die MIRIAM-Identifikatoren ins Leben gerufen. Durch die Implementierung dieser in MonaLisa, und durch die Unterstützung des SBML-Dateiformats, wird es möglich, die erstellten Modelle für eine Veröffentlichung in einer Fachzeitschrift oder einer Datenbank, wie Biomodels (Li et al., 2010), vorzubereiten. Da auch vorhandene Annotationen aus SBML-Dateien ausgelesen werden können, unterstützt diese Funktionalität ebenfalls

das bessere Verständnis von fremden Modellen. Die Annotation eines Elements mit einem SBO-Begriff führt zu dessen semantischer Interpretation. Gerade in komplexeren Modellen oder bei Modellen von Signaltransduktionswegen, bei denen die klare Aufteilung zwischen Proteinen und Metaboliten als Transitionen und Plätzen fehlt, ist eine solche Einordnung für das Verständnis des Modells sehr hilfreich.

## 3.5 Der *TreeViewer*

Neben dem *NetViewer* existiert eine zweite Visualisierungkomponente, der *TreeViewer*. Der *TreeViewer* ist in Zusammenarbeit mit Stefan Marchi entstanden und dient zur Visualisierung der T-Cluster. Der *TreeViewer* verwendet, wie der *NetViewer*, die JUNG-Bibliothek zur Visualisierung der berechneten Clusterbäume. Abbildung 3.30 zeigt die grafische Oberfläche des *TreeViewer*. Im linken Teil ist der Ausschnitt eines Clusterbaumes zu sehen. Die einzelnen Knoten repräsentieren die berechneten Cluster und die Länge der Kante spiegelt ihre Distanz zur Wurzel wider. Wird ein Knoten ausgewählt, so wird

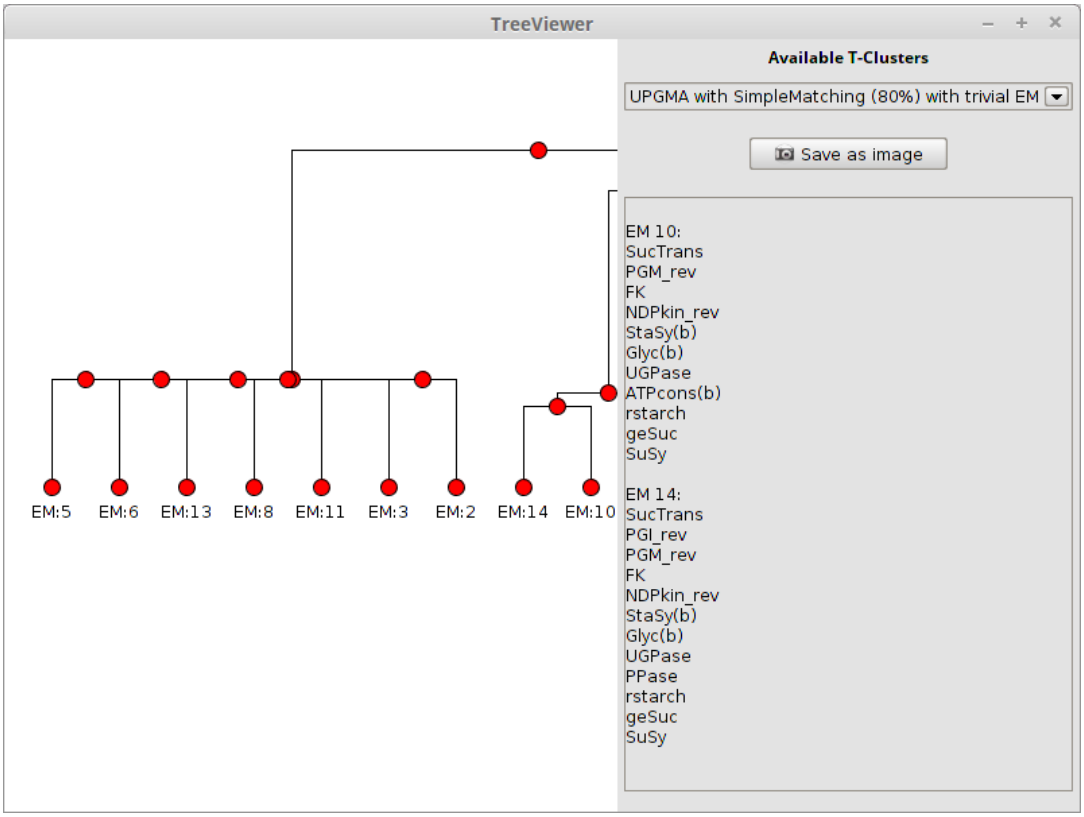


**Abbildung 3.30:** Die Abbildung gibt eine Übersicht über den *TreeViewer*. Im linken Teil befindet sich die Visualisierung des Clusterbaumes. Knoten repräsentieren einen der berechneten Cluster. Wird ein solcher ausgewählt, so wird im rechten Bereich eine Übersicht gegeben, aus welchen T-Invarianten (EM) sich dieser zusammensetzt und welche Transitionen in diesem enthalten sind. Über die Schaltfläche *Save as Image* kann die aktuelle Ansicht des Clusterbaumes als Bild gespeichtert werden. Darüber befindet sich die Auswahlliste aller verfügbaren Clusterbäume. Der hier ausgewählte Eintrag wird dann im linken Bereich visualisiert.

im rechten Bereich eine Liste der enthaltenen T-Invarianten (EM) angezeigt sowie die dazugehörigen Transitionen. Über diesem Bereich befindet sich die Schaltfläche *Save as Image*, mit der die aktuelle Ansicht des Clusterbaumes als Bild gespeichert werden kann. Zur Auswahl steht hierbei das PNG- oder das SVG-Format. Darüber befindet sich die Auswahlliste mit allen verfügbaren Clusterbäumen. Wird in dieser Liste ein Eintrag ausgewählt, so wird dieser im linken Bereich angezeigt.

Das Ergebnis der T-Cluster ist eine Baumstruktur und kann als solche nicht direkt auf das PN im *NetViewer* projiziert werden. Daher steht mit dem *TreeViewer* eine gesonderte Komponente bereit, um die Resultate der T-Cluster-Analyse zu visualisieren. Aktuell fehlt jedoch eine Verbindung zwischen dem *TreeViewer* und dem *NetViewer*, um zum Beispiel alle Transitionen eines Clusters farblich hervorzuheben oder zwei Cluster miteinander vergleichen zu können. Eine solche Interaktion zwischen den beiden Visualisierungskomponenten würde eine noch bessere Interpretation und Analyse der T-Cluster ermöglichen und ist für die Weiterentwicklung von MonaLisa ein nächster Schritt.

## 3.6 Vergleich mit anderen Programmen

In Tabelle 3.3 wird ein Vergleich von MonaLisa mit anderen Programmen zur Modellierung und Analyse von PN gezeigt. Die Auswahl ist hier auf Programme beschränkt, die eine direkte PN-Modellierung im biologischen Kontext ermöglichen:Vanted (Rohn et al., 2012) mit der Erweiterung Petri-Net (Hartmann et al., 2012) und Snoopy (Fieber, 2004). Im Unterschied zu Snoopy sind Vanted und MonaLisa in Java implementiert

| Funktion | MonaLisa | Vanted | Snoopy |
|---|---|---|---|
| Verfügbarkeit | offen | offen | geschlossen |
| Software-Plattform | Java | Java | C++ |
| PN-Varianten | P/T-Systeme | P/T-Systeme | P/T-Systeme und viele Weitere |
| T-Invarianten | ✓ | ✓ | – |
| P-Invarianten | ✓ | ✓ | – |
| MCT-Sets | ✓ | – | – |
| Knock-out-Analyse | ✓ | – | – |
| T-Cluster | ✓ | – | – |
| MCS | ✓ | – | – |
| deterministische Simulation | ✓ | ✓ | ✓ |
| stochastische Simulation | ✓ | – | ✓ |
| topologische Analysen | ✓ | – | – |
| Visualisierung der Resultate | ✓ | ✓ | – |
| SBML Unterstützung | ✓ | ✓ | nur bis Level 2 Version 4 |
| Annotaionen | ✓ | – | – |

**Tabelle 3.3:** Die Tabelle zeigt den Vergleich von MonaLisa mit weiteren Programmen zur Modellierung und Analyse von PN. Die Auswahl der Programme ist auf solche beschränkt, die in einem biologischen Kontext stehen.

und quelloffen. Dies ermöglicht es Dritten für diese beiden Programme neue Funktionen zu entwickeln und zu implementieren. Sowohl VANTED, als auch MONALISA stellen dafür spezielle Klassenstrukturen bereit. SNOOPY hat sich darauf spezialisiert möglichst viele PN-Varianten zu unterstützen. MONALISA und VANTED unterstützen hingegen nur klassische P/T-Systeme. Mit SNOOPY kann das PN nur modelliert und simuliert werden, es stehen jedoch keine weiteren Analysemethoden zur Verfügung. Im Vergleich zu VANTED, stellt MONALISA ein breiteres Spektrum an Analysemethoden für PN zur Verfügung. In allen drei Anwendungen ist die Simulation eines PN möglich, jedoch in unterschiedlichen Ausprägungen und unterschiedlichem Umfang, wobei in VANTED nur eine deterministische Simulation möglich ist. Sowohl VANTED als auch MONALISA wurden für einen Einsatz in der Biologie entwickelt und verwenden daher entsprechende Terminologie und stellen Funktionen speziell für die Biologie zur Verfügung. SNOOPY orientiert sich dahingegen mehr an technischen Anwendungen. MONALISA stellt weitaus mehr Analysemethoden für die systembiologische Untersuchung von PN zur Verfügung. Eine Visualisierung der Resultate der Analysemethoden ist in den beiden Anwendungen VANTED, im Bezug auf die PN - Analyse, und SNOOPY nur sehr eingeschränkt möglich. Auch hier bietet MONALISA den größeren Funktionsumfang.

Zusammenfassend lässt sich sagen, dass die Stärken von SNOOPY die ist, dass die Modellierung nicht nur mit klassischen P/T-Systeme möglich ist. Für weitergehende Analysen des PN muss dann jedoch auf weitere Programme zurückgegriffen werden, so unterstützt beispielsweise MONALISA auch das Dateiformat von SNOOPY. Die Stärke von VANTED ist, dass experimentelle Daten auf das PN visualisiert werden können. Für die Modellierung eines biologischen Systems mit Hilfe von PN, und anschließender ausgiebiger Analyse dieses Modells, ist MONALISA die beste Wahl aus diesen drei Anwendungen. Durch die Unterstützung von SBML und anderen Dateiformaten ist ein Austausch zwischen diesen Anwendungen jedoch jederzeit möglich.

# Kapitel 4

# Zusammenfassung und Schlussfolgerung

Die mathematische Modellierung biologischer Systeme ist ein wichtiger Bestandteil der Systembiologie. Mit Hilfe mathematischer Modelle lässt sich das Wissen der verschiedensten Hochdurchsatz-Methoden, und der durch diese erzeugten *-om*-Daten, zusammenführen und ermöglicht ein Studium des biologischen Systems auch außerhalb des Labors. (Kitano, 2002)

Für größere biologische Systeme stehen jedoch meistens nicht alle Informationen über Stoffkonzentrationen oder Reaktionsgeschwindigkeiten zur Verfügung, um eine quantitative Modellierung durchführen zu können. In einem solchen Fall wird auf Methoden der qualitativen Modellierung zurückgegriffen. Eine dieser Methoden sind die PN, welche seit Anfang der 1990er Jahre Verwendung in der Systembiologie finden, um zum Beispiel metabolische Systeme oder Signaltransduktionswege zu modellieren. Einer der Vorteile dieser Methode ist, dass Modelle als qualitative Beschreibung des Systems begonnen werden können und im Laufe der Zeit um quantitative Beschreibungen ergänzt werden können. Zur Modellierung von PN existieren bereits viele Anwendungen. Da das Konzept der PN jedoch ursprünglich nicht für die Systembiologie entwickelt wurde und meist im technischen Bereich verwendet wird, existierten kaum Anwendungen, die für den Einsatz in der Systembiologie entwickelt wurden. Daher ist auch die Durchführung der für die Systembiologie entwickelten Analysemethoden für PN nicht mit diesen Anwendungen möglich. Die Motivation des ersten Teiles dieser Arbeit war es daher, eine Anwendung zu schaffen, die speziell für die PN-Modellierung und Analyse in der Systembiologie gedacht ist, also sich in ihren Analysemethoden und ihrer Terminologie an den Bedürfnissen der Systembiologie orientiert.

Aus dieser Motivation heraus wurde die Software MonaLisa entwickelt, welche im ersten Teil dieser Arbeit vorgestellt wurde. MonaLisa ist in Java 1.7 implementiert, der Quelltext ist offen und unter der *Artistic Licence 2.0* frei verfügbar. Mit MonaLisa können Modelle biologischer Systeme als PN modelliert und analysiert werden. Die

Resultate der verschiedenen Analysemethoden können direkt auf das PN visualisiert werden, was den Modellierer bei der Auswertung und Interpretation der Resultate unterstützt. Da die Anwendung viele verschiedene Analysemethoden zur Verfügung stellt, war es nötig, Strukturen zu schaffen, mit denen diese Analysemethoden einheitlich in MonaLisa integriert werden können. Dazu zählen Strukturen für die Analysemethoden selbst, aber auch zum Speichern ihrer Resultate und der verwendeten Parameter. Die Visualisierung und Modellierung des PN findet in einer eigenständigen Komponente von MonaLisa statt, dem *NetViewer*. Damit das erstellte PN und die vorhandenen Resultate gespeichert und zu einem späteren Zeitpunkt wieder verwendet werden können, ist es möglich, diese Informationen in einer Projektdatei zu speichern. Zudem wurden Klassen für den Import und Export von externen Dateiformaten implementiert, wie zum Beispiel von SBML, KGML und PNT. Dies ermöglicht den Austausch der Modelle mit anderen Programmen, wie beispielsweise Cytoscape oder CellDesigner, oder Datenbanken, wie beispielsweise MetaCyc oder Biomoldes, die in der Systembiologie und in der PN-Gemeinde Standards darstellen.

MonaLisa ermöglicht, noch vor der Untersuchung struktureller oder dynamischer Eigenschaften, die Untersuchung der elementaren Netzwerkeigenschaften. So kann beispielsweise die Homogenität, die Reinheit oder die statische Konfliktfreiheit des PN überprüft werden. Ein Großteil der Analysemethoden beschäftigt sich mit den strukturellen Eigenschaften des PN, sie sind also nur von der Topologie des PN abhängig. Mit den Transitions- und Platz-Invarianten können funktionelle Einheiten im PN identifiziert werden, wie zum Beispiel Gruppen von Enzymen im Fließgleichgewicht beziehungsweise eine Stoffkonservierung. Diese funktionellen Einheiten können anschließend reduziert und eingeschränkt werden, wenn zum Beispiel die Zahl der T-Invarianten zu groß ist. Die gefundenen funktionellen Einheiten können jeweils im PN angezeigt und farblich hervorgehoben werden. Weitere Methoden helfen den Ausfall einer Reaktion oder eines Metaboliten zu untersuchen. Die Topologie des PN kann mit MonaLisa ebenso untersucht werde wie die Berechnung verschiedener Zentralitätsmaße zum *Gene-Ranking*. Neben den strukturellen Eigenschaften des PN kann mit MonaLisa auch das dynamische Verhalten des PN untersucht werden. Dies kann deterministisch oder mit Hilfe stochastischer Algorithmen erfolgen. MonaLisa ermöglicht auch die Annotation des PN sowie dessen Elemente mit MIRIAM-Identifikatoren und SBO-Begriffen.

## 4.1 Visualisierung der Resultate der Analysemethoden

Die in dieser Arbeit vorgestellten und in MonaLisa integrierten Analysemethoden ermöglichen eine ausgiebige Untersuchung von PN-Modellen. MonaLisa unterstützt die Auswertung der Resultate dieser Analysemethoden durch die Visualisierung dieser auf das PN. Dadurch wird es möglich, die Resultate direkt im Kontext des PN zu betrachten und sie biologisch zu interpretieren, anstatt die Textausgabe der Analysemethoden manuell auf das PN zu übertragen. Eine solche Darstellung der Resultate wird zudem enorm wichtig, je komplexer das PN-Modell wird, denn in einem solchen Fall steigt auch die Größe

der Resultate einzelner Analysemethoden (Klamt und Stelling, 2002) und eine manuelle Zuordnung auf die Elemente des PN wird kaum durchführbar. Durch die Visualisierung der Resultate auf das PN kann auch eine größere Menge an Resultaten nach biologisch sinnvollen Aussagen, wie beispielsweise biochemische Module innerhalb des Modells oder alternative Wege eines Signaltransduktionsweges, durchsucht und Fehler schnell entdeckt werden. Für die Entwicklung von MonaLisa war dieses Konzept ein wichtiges Kriterium. Zu jeder implementierten Analysemethode wurde daher zusätzlich eine Möglichkeit geschaffen, deren Resultate, sofern möglich, grafisch darzustellen. Ein Konzept, welches für die weitere Entwicklung von MonaLisa für jede neue Analysemethode ebenfalls umgesetzt werden sollte.

## 4.2 Eine Anwendung speziell für die Systembiologie

Für den Einsatz von MonaLisa speziell in der Systembiologie wurden gezielt solche Analysemethoden integriert, die in der Systembiologie entstanden sind oder für biologische Systeme wichtig sind. Hierzu zählen zum Beispiel die *Maximal Common Transition Sets* oder T-Cluster zur weiteren Untersuchung der T-Invarianten. Die Knock-out-Analyse und die *Minimal Cut Sets* sind beides wichtige Bestandteile einer systembiologischen Analyse eines PN. Für die Simulation des dynamischen Verhaltens eines PN steht unter anderem der Algorithmus von Gillespie zur Simulation eines chemischen Systems zur Verfügung, zum Beispiel eines metabolischen Modells. Neben den Analysemethoden spiegelt sich dies aber auch in den weiteren vorhandenen Funktionalitäten wieder. So ist zum Beispiel ein automatisiertes Anlegen reversibler Reaktionen oder das Anlegen verschiedener Zellkompartimente möglich. Durch die Annotation des Modells und seiner Elemente ist eine eindeutige Identifikation dieser möglich und ein Austausch mit Dritten wird erleichtert. Zuletzt ist die in MonaLisa verwendete Terminologie auf die Systembiologie abgestimmt. So werden zum Beispiel Plätze als *Species* und Transitionen als *Reactions* benannt, Bezeichnungen, die so auch im SBML-Format zum Einsatz kommen und T-Invarianten als Elementarmoden bezeichnet, ein in der Systembiologie geläufigerer Begriff.

## 4.3 Flexibilität von MonaLisa

Für die PN-Modellierung existieren bereits viele Analysemethoden und diese werden durch immer neue Ansätze ergänzt. Zur Integration all dieser, schon existierenden aber auch zukünftigen Methoden, ist es notwendig, dass MonaLisa Strukturen bereitstellt, mit deren Hilfe diese Analysemethoden leicht und einheitlich in die Struktur der Anwendung eingebaut werden können. Zu diesem Zweck wurden für MonaLisa Programmstrukturen geschaffen, die eine einheitliche Behandlung der Analysemethoden, sowie deren Parametern und Resultaten ermöglichen. Neben der Durchführung der Analysemethoden ist auch das Abspeichern und Laden der Resultate einheitlich geregelt. Der Zugriff auf zentrale Datenstrukturen von MonaLisa, wie zum Beispiel der Struktur zur Repräsentation des

PN, ist ebenfalls durch diese Strukturen geregelt und für alle Analysemethoden gleich. Zudem erhalten die Analysemethoden auch Zugriff auf die Visualisierung des PN und können somit in allen Ebenen von MonaLisa eingreifen und neue Funktionen bereitstellen. Durch diese Modularität ist es möglich, schnell und ohne großen Eingriff in vorhandene Strukturen, neue Funktionalitäten zu MonaLisa hinzuzufügen. Der offene Quelltext und der Einsatz der *Artistic Licence 2.0* ermöglicht zudem Dritten, den Funktionsumfang von MonaLisa zu erweitern.

# Kapitel 5

# Anhang

In den folgenden Abschnitten werden Dateiformate beschrieben. Die dazugehörige Syntax wird in Tabelle 5.1 erläutert.

| Symbol | Bedeutung |
|---|---|
| <Foo> | Platzhalter für ein Element, zum Beispiel den Namen einer Reaktion |
| ' + ' | Eine definierte Zeichenfolge im Dateiformat. Leerzeichen innerhalb dieser Zeichenfolge werden beachtet. |
| [ ] | Alle Elemente zwischen diesen Zeichen kommen mindestens einmal vor, können jedoch beliebig oft wiederholt werden. |
| { } | Kann nur innerhalb einer Wiederholung vorkommen. Elemente zwischen diesem Symbol werden bei der letzten Wiederholung ignoriert. |
| \| | Zeilenumbruch. |
| ( ) ^ ( ) | Alternative. Alternativen werden durch die Elemente innerhalb der jeweiligen Klammer gegeben. Es sind mehr als zwei Alternativen möglich. |

**Tabelle 5.1:** Die Tabelle gibt einen Überblick über die zum Beschreiben eines Dateiformates verwendete Syntax.

## 5.1 Das *Plain*-Dateiformat

Dateien dieses Formates beinhalten eine Liste aller im PN vorhandenen Reaktionen. Es folgt dabei dem Prinzip der Annotation chemischer Reaktionen:

```
[<Reaktionsname> ' : ' [<Kantengewicht>'*'<Vorplatz> {' + '} ]
' -> ' [<Factor>'*'<Nachplatz> {' + '}] { | } ]
```

Sollte das Kantengewicht 1 betragen, so wird dies nicht aufgeführt. Die Endung der Dateien ist *txt*.

Ein Beispiel ist:

```
SucTrans: eSuc -> Suc
Inv: Suc -> Frc + Glc
HK: Glc + ATP -> ADP + G6P
FK: Frc + ATP -> ADP + F6P
SPP: S6P -> Suc + Pi
StaSy(b): G6P + ATP -> starch + ADP + PP
Glyc(b): F6P + 29*ADP + 28*Pi -> 29*ATP
```

## 5.2 Dateiformat zum Export von T- und P-Invarianten

Das Dateiformat zum Export von T-Invarianten orientiert sich am PNT-Format (siehe Abschnitt 2.7.2). Die beim Export erstellten Dateien haben die Endung *inv*. Am Anfang der Datei steht eine Liste aller Transitionen des modellierten Systems, wobei jeder Reaktion ein eindeutiger Identifikator zugewiesen wird. Dieser Identifikator ist eine Zahl, startet bei 1 und wird fortlaufend erhöht. Die Identifikatoren werden benötigt, um in der nachfolgenden Auflistung aller T-Invarianten ein eindeutiges Format zu erhalten. Würden hier die Namen der Reaktionen verwendet werden, könnte kein eindeutiges Trennzeichen zwischen den einzelnen Transitionen einer T-Invariante definiert werden, da mögliche Trennzeichen ebenfalls im Namen vorkommen könnten.

```
'# reaction_id:name' |
[<TransitionID>':'<Transitionname> | ] |

'# em_id:factor*reaction_id; ...' |
[ <TInvariantID>':'[ <Faktor>'*'<TransitionID> { ; } ] { | } ]
```

Ein Beispiel ist:

```
# reaction_id:name
1:Input
2:Output
3:Reaction_1_reversible
4:Reaction_1
5:Reaction;B->C
6:Reaction;B->C_reversible
7:Output_2

# em_id:factor*reaction_id; ...
1:3*1;1*2;3*3;6*5;6*7;
2:3*1;1*2;3*4;
3:3*1;1*2;3*3;6*6;6*7;
```

Das Dateiformat für den Export von P-Invarianten ist analog zu diesem Format. Die Aufzählung aller Transitionen wird zu einer Aufzählung aller Plätze. Die Liste aller P-Invarianten besteht hier aus den enthaltenen Plätzen und deren Faktoren. Die Endung der Datei ist ebenfalls *inv*.

## 5.3 Dateiformat zum Export von MCT-Sets

Das Dateiformat zum Export von MCT-Sets basiert auf dem Dateiformat zum Export von T-Invarianten (siehe 5.2). Die beim Export erstellten Dateien haben die Endung *mcts*.

```
'# reaction_id:name' |
[<TransitionID>':'<Transitionname> | ] |

'# mcts_id:reaction_id; ...' |
[ <MctsID>':'[ <TransitionID> { ; } ] { | } ]
```

Ein Beispiel:

```
# reaction_id:name
1:Input
2:Output
3:Reaction_1_reversible
4:Reaction_1
5:Reaction;B->C
6:Reaction;B->C_reversible
7:Output_2

# mcts_id:reaction_id; ...
1:1;2;
2:3;7;
```

## 5.4 Dateiformat zum Export von Knock-out-Analysen

Das Dateiformat zum Export der Knock-out-Analysen folgt ebenfalls dem Prinzip des PNT-Dateiformates (siehe Abschnitt 2.7.2). Zu Beginn der Dateien, welche die Endung *txt* haben, erfolgt eine Auflistung aller Plätze und Transitionen des PN und eine Zuordnung zu einem eindeutigen Identifikator. Es schließt sich eine Aufzählung der durchgeführten Knock-out-Analysen an. Hier muss unterschieden werden, ob eine Transition oder ein Platz ausgeschaltet wurde.

```
'# reaction_id:name' |
[<TransitionID>':'<Transitionname> | ] |

'# species_id:name' |
[<TransitionID>':'<Transitionname> | ] |

(# ko_reactions_id; ... : ko_affected_reactions_id; ...) ^
(# ko_species_id; ... : ko_affected_reactions_id; ...) |

[ [ (KnockedOutReactionID) ^ (<KnockedOutSpeciesID>) {;}  ] ':'
[<AffectedReactionID {;} ] { | } ]
```

Ein Beispiel eines Ausschnittes:

```
# reaction_id:name
1:SucTrans
2:Inv
3:HK
4:FK
.
.

# species_id:name
1:Suc
2:eSuc
3:Glc
4:Frc
5:UDPglc
.
.

# knocked_out_reaction_id; ... : also_knocked_out_species_id; ...
10;3:1;2;4;5;6;7;8;9;24
```

## 5.5 Dateiformat zum Export von MCS

Das Dateiformat zum Export von MCS beinhaltet zuerst eine Liste aller Transitionen des PN und eine Zuordnung dieser zu einem eindeutigen Identifikator. Es schließt sich eine Liste aller gefundenen MCS an, welche ebenfalls einen eindeutigen Identifikator zugewiesen bekommen. Dateien mit diesem Format haben die Endung *mcs*.

```
'# reaction_id:name' |
[<TransitionID>':'<Transitionname> | ] |

'# mcs_id:reaction_id; ...' |
[ <MctsID>':'[ <TransitionID> { ; } ] { | } ]
```

# Teil II

# Topologische Analyse biochemischer Netzwerke

# Kapitel 1

# Einleitung

Das Wissen und Verständnis über die molekularen Prozesse des Lebens ist in den letzten Jahrzehnten enorm gewachsen (Winterbach et al., 2013). Dies wurde unter anderem durch die Entwicklung neuer Methoden möglich, mit denen zum Beispiel die Konzentration oder die Anwesenheit bestimmter Stoffe, auch genomweit, gemessen werden kann. Hierzu zählen beispielsweise *Microarrays* (Kononen et al., 1998; MacBeath und Schreiber, 2000) oder die Massenspektrometrie (Fenn et al., 1989; Aebersold und Mann, 2003). Besonders für die Systembiologie, dem Bereich der Biologie, der biologische Systeme in ihrer Gesamtheit erforscht und untersucht, ist dieser Wissenszuwachs wichtig. Für Systembiologen ist jedoch neben dem Wissen über alle vorhandenen Proteine und Gene auch die Interaktion zwischen diesen von großem Interesse, weshalb auch hierfür verschiedenste Methoden entwickelt wurden, wie zum Beispiel das Hefe-Zwei-Hybrid-System (Fields und Song, 1989; Ito et al., 2001) oder die *Tandem Affinity Purification* (Puig et al., 2001). Mit Hilfe dieser Methoden können Interaktome, also die Gesamtheit der molekularen Interaktionen eines biologischen Systems, erstellt werden (Cusick et al., 2005; De Las Rivas und Fontanillo, 2010). Das so gewonnene Wissen wird in Datenbanken zusammengetragen und ist somit für andere Wissenschaftler zugänglich. Beispiele für solche Datenbanken sind KEGG (Kanehisa et al., 2014), BRENDA (Scheer et al., 2010) oder MetaCyc (Caspi et al., 2008).

Das Wissen über Interaktome ist auch die Grundlage vieler computergestützter Methoden zur Analyse, Modellierung, Interpretation und Vorhersage biologischer Phänomene (Cho et al., 2012; Winterbach et al., 2013). Die Interaktionen zischen Proteinen oder Genen werden dabei oft als Netzwerk oder Graph modelliert, ein Ansatz der schon lange existiert (Rashevsky, 1954) und immer weiter entwickelt wurde (Ideker und Lauffenburger, 2003). Das mathematische Konzept der Graphentheorie wurde schon im 18. Jahrhundert entwickelt und definiert einen Graphen als eine abstrakte Struktur, mit der eine Menge von Objekten sowie deren Beziehung zueinander darstellt werden kann (West, 2001). Eine der frühen Anwendungen der Graphentheorie war die Abbildung chemischer Strukturen (Sylvester, 1878) oder Konstitutionsformeln chemischer Verbindungen(Cayley, 1874). Übertragen auf die Interaktome, repräsentiert ein Graph die vorhandenen Proteine

oder Gene und die physikalischen Beziehungen dieser untereinander. Es existieren unterschiedlichste Ausprägungen eines solchen Graphen für die Anwendung in der Biologie. In einem Hypergraphen werden zum Beispiel Metaboliten durch Knoten repräsentiert, die mit Kanten verbunden sind, welche die Enzym-katalysierten Reaktionen darstellen, durch welche die Metaboliten ineinander umgewandelt werden (Arita, 2004). Wird dieses Konzept ebenfalls auf die Reaktionen eines biologischen Systems angewendet, so erhält man den Reaktionen-Graph, in welchem die Knoten die Enzym-katalysierten Reaktionen repräsentieren, die durch Kanten verbunden werden, wenn zwei Reaktionen sich Metaboliten teilen (Wagner und Fell, 2001). Sollen Metaboliten und Reaktionen im gleichen Graph modelliert werden, kann auf den Petrinetz-Formalismus (PN) (Petri, 1962; Murata, 1989; Koch et al., 2011) zurückgegriffen werden. Ein PN ist ein gerichteter, bipartiter Graph, dessen beiden disjunkten Knotenmengen in einem metabolischen Modell jeweils die Metaboliten und die Enzym-katalysierten Reaktionen repräsentieren (Reddy et al., 1993; Nöthen, 2014). Die Kombination von Systembiologie, Graphentheorie und die statistische Analyse der Topologie dieser Graphen wird Netzwerkbiologie genannt (Barabasi und Oltvai, 2004).

Die Analyse der topologischen Eigenschaften eines Graphen ermöglicht es, grundlegende Aussagen über die globalen Eigenschaften des modellierten Systems und dessen Entstehungsprozesses zu treffen. Daher ist eine solche Analyse oft der erste Schritt für das Verständnis eines komplexen biologischen Systems. In der Arbeit von Jeong et al. (2000) wurde die Topologie des Metabolismus von 43 verschiedenen Organismen untersucht. Die Autoren kamen zu dem Schluss, dass die Verteilung der Knotengrade skalenfrei ist, also einem Potenzgesetz folgt (Barabás und Bonabeau, 2003). Dies bedeutet, dass einige wenige Metaboliten mit sehr hohem Knotengrad existieren und so viele andere Metaboliten miteinander verbinden. Dadurch wiederum ist das metabolische System sehr robust gegenüber dem Ausfall von Metaboliten (Albert et al., 2000). Für Reaktions-Graphen wurde ebenfalls eine skalenfreie Verteilung der Knotengrade festgestellt (Wagner und Fell, 2001). Das Verhältnis der Größe eines metabolischen Systems und dem Knotengrad der einzelnen Metaboliten wurde in der Arbeit von Albert und Barabási (2002) beleuchtet, welche zu dem Ergebnis kamen, dass in größeren metabolischen Systemen die Metaboliten jeweils auch an einer größeren Zahl von unterschiedlichen Reaktionen beteiligt sind. Für die Verteilung des Clusterkoeffizienten, einem Maß für den Grad der Vernetzung der direkten Umgebung eines Knotens, wurde ebenfalls eine Skalenfreiheit festgestellt (Ravasz et al., 2002), zudem wurde hier gezeigt, dass der Clusterkoeffizient der einzelnen Knoten nicht von der Größe des Systems abhängig ist. In der Arbeit von Hao et al. (2012) wurde der Zusammenhang einer skalenfreien Verteilung des Knotengrades der Metaboliten und der skalenfreien Verteilung des Clusterkoeffizienten dieser aufgezeigt.

## 1.1 Motivation

Für das Verständnis dieser komplexen biologischen Systeme ist die Analyse deren topologischen Eigenschaften ein erster Schritt. In den bisherigen Arbeiten zur Untersuchung der topologischen Eigenschaften von metabolischen Systemen wurden diese nur für Metaboliten und Reaktionen in getrennten, oft ungerichteten, Graphen durchgeführt. Die Motivation dieser Arbeit ist es, durch den Einsatz des PN-Formalismus, Metaboliten und Reaktionen in einem Graph zusammenzufassen. So wird es möglich, die topologischen Eigenschaften für beide Elemente in einer gemeinsamen Topologie zu untersuchen, die sowohl die Stöchiometrie als auch Reversibilität der Reaktionen berücksichtigt. Neben der Untersuchung der topologischen Eigenschaften der Reaktionen soll ebenfalls untersucht werden, ob die schon beschriebenen topologischen Eigenschaften für Metaboliten bei einer solchen Repräsentation erhalten bleiben. Die Analyse der topologischen Eigenschaften wird oft auf das Modell eines Organismus beschränkt. Für diese Arbeit wurde die 2641 Gesamtgenom-Modelle aus der *path2models*-Datenbank untersucht, um einen Überblick über möglichst viele Organismen zu erhalten. Diese Arbeit betrachtet zwei topologischen Eigenschaften des Knotengrades und des Clusterkoeffizienten. Die Ergebnisse sollen hierbei als Grundlage für weiterführende Analysen, welche die hier gefundenen Eigenschaften und Phänomene gründlicher untersuchen, dienen. Ein Abstecken dieser Eigenschaften für metabolische Modelle kann in Zukunft auch helfen, neu erstellte Modelle in einem ersten Schritt zu validieren.

## 1.2 Aufbau der Arbeit

Zunächst wird auf die Implementierung der Anwendung und der Datenbank eingegangen, welche für die Durchführung der topologischen Analysen entwickelt wurde. Im Anschluss wird die Umwandlung der verwendeten Modelle in PN-Modelle beschrieben. Darauf folgt die Betrachtung des Verhältnisses der Anzahl der Reaktionen zur Anzahl der Metaboliten in den Modellen, gefolgt von den Ergebnissen der Untersuchung des Knotengrades und des Clusterkoeffizienten. Im Anschluss an dieses Kapitel folgt eine kurze Zusammenfassung der Ergebnisse.

# Kapitel 2

# Material und Methoden

## 2.1 Reaktionssysteme und Graphen

Die Struktur eines biochemischen Netzwerkes kann durch einen bipartiten Graph beschrieben werden, dessen Elemente jeweils die Metaboliten und die Reaktionen dieses repräsentieren. Hierzu kann der PN-Formalismus (Petri, 1962; Murata, 1989; Koch et al., 2011) verwendet werden, welcher im Abschnitt 2.1 des ersten Teiles dieser Arbeit eingeführt wurde. Die folgenden Abschnitte basieren auf den dort eingeführten Definitionen.

## 2.2 Die Nachbarschaft eines Knotens

Die Vorknoten und Nachknoten eines Knotens $n_i$ definieren seine Nachbarschaft 1. Grades $V_{\mathrm{G}}^{(1)}(n_i) = (\bullet n_i) \cup (n_i \bullet)$. Für einen gegeben Knoten $n \in V = T \cup P$ eines gerichteten, bipartiten Graphen $G = (P, T, E)$ ist die Menge der Nachbarn 2. Grades $V_{\mathrm{G}}^{(2)}(n)$ definiert durch

$$V_{\mathrm{G}}^{(2)}(n) \ = \{n' \in V \mid \exists\, x \in V : \ ((n', x), (x, n) \in E) \ \vee \ ((n, x), (x, n') \in E)\} \ . \qquad (2.1)$$

Die Nachbarschaft 1. Grades einer Transition $t \in T$ beinhaltet nur Plätze, niemals Transitionen, $V_{\mathrm{G}}^{(1)}(t) \subset P$. Die Nachbarschaft 2. Grades einer Transition $t \in T$ beinhaltet nur Transitionen, niemals Plätze, $V_{\mathrm{G}}^{(2)}(t) \subset T$. Gleiches gilt für Plätze: $p \in P \ \Rightarrow \ V_{\mathrm{G}}^{(1)}(p) \subset T \ \wedge \ V_{\mathrm{G}}^{(2)}(p) \subset P$. Es existiert immer ein gerichteter Pfad zwischen einem Knoten $n$ und seinen Nachbarn. Die Länge dieses gerichteten Pfades entspricht 1 zwischen einem Knoten $n$ und seinen Nachbarn 1. Grades und hat eine Länge von 2 zwischen $n$ und seinen Nachbarn 2. Grades. In einem biochemischen Netzwerk entspricht die Menge der Nachbarn 1. Grades einer Transition $t \in T$ den Edukten und Produkten einer Reaktion. Für einen Metaboliten entspricht die Menge der Nachbarn 1. Grades eines Platzes $p \in P$ den Reaktionen, an welchen es als Produkt oder Edukt teilnimmt.

## 2.3 Knotengrad

Der Knotengrad $k_i \equiv |V_{\mathrm{G}}^{(1)}(i)|$ in einem gerichteten Graphen entspricht der Anzahl der Kanten von und zu einem Knoten $n_j$. Es wird zwischen dem ausgehenden und dem eingehenden Knotengrad unterschieden. $k_{n_j}^o \equiv |n_j \bullet|$ ist die Anzahl der Kanten, welche am Knoten $n_j$ starten und $k_{n_j}^i \equiv |\bullet\, n_j|$ ist die Anzahl der Kanten, welche am Knoten $n_j$ enden. Die biologische Interpretation des Knotengrades hängt davon ab, welcher Knotentyp betrachtet wird. Eine Transition $t \in T$ repräsentiert eine Reaktion in einem biochemischen Netzwerk. Daher entspricht der ausgehende Knotengrad $k_t^o$ der Anzahl der von Reaktion $t$ erzeugten Metaboliten. Die Anzahl der konsumierten Metaboliten ist dementsprechend durch $k_t^i$ gegeben. Ein Platz $p \in P$ repräsentiert einen Metaboliten in einem biochemischen Netzwerk. Der ausgehende Knotengrad $k_p^o$ entspricht der Anzahl der Reaktionen welche diesen Metaboliten konsumieren. Die Anzahl der Reaktionen, welche diesen Metaboliten $p$ produzieren, wird durch $k_p^i$ gegeben.

## 2.4 Clusterkoeffizient

Der Clusterkoeffizient des Knotens $n$ in einem PN wird durch den Quotient

$$C_n = \frac{m_n}{q_n(q_n - 1)} \tag{2.2}$$

definiert. Der Zähler $m_n$ zählt die Anzahl von verbundenen Paaren in der Nachbarschaft 2. Grades $V_{\mathrm{G}}^{(2)}(n)$ des Knoten $n$. Da alle Knoten in $V_{\mathrm{G}}^{(2)}(n)$ aus der gleichen Knotenmenge wie $n$ stammen, sind diese niemals direkt verbunden. Wir nennen ein geordnetes Knotenpaar $(n, n')$ verbunden, falls ein gerichteter Pfad der Länge 2 von $n$ zu $n'$ existiert. Eine Formel für die Berechnung von $m_n$ ist:

$$\begin{aligned} m_n = \frac{1}{2} \sum_{l \in V_{\mathrm{G}}^{(2)}(n)} & \mid \{k \in (V_{\mathrm{G}}^{(2)}(l) \ \cap \ V_{\mathrm{G}}^{(2)}(n)) \mid \exists\, x \in V : (l,x),(x,k) \in E\} \mid \\ & + \mid \{k \in (V_{\mathrm{G}}^{(2)}(l) \ \cap \ V_{\mathrm{G}}^{(2)}(n)) \mid \exists\, x \in V : (k,x),(x,l) \in E\} \mid . \end{aligned} \tag{2.3}$$

Ein Knotenpaar $(n, n')$ mit einem gerichteten Pfad $n \to n'$ und einem gerichteten Pfad $n' \to n$ trägt zwei gerichtete Knotenpaare zu $m_n$ bei. Der Nenner $q_n(q_n - 1)$ in Definition 2.2 entspricht der Anzahl aller möglichen geordneten Knotenpaare in $V_{\mathrm{G}}^{(2)}(n)$, wobei $q_n = \mid V_{\mathrm{G}}^{(2)}(n) \mid$ entspricht. Der Clusterkoeffizient kann einen Wert zwischen 0 und 1 annehmen. Bei einem Wert von $C_n = 0$ ist kein geordnetes Knotenpaar in $V_{\mathrm{G}}^{(2)}(n)$ vorhanden. Ein Wert von $C_n = 1$ wird erreicht, wenn alle Hin- und Rückwege zwischen allen Knoten in $V_{\mathrm{G}}^{(2)}(n)$ vorhanden sind.

## 2.5 Skalenfreie Netzwerke

Ein Netzwerk wird skalenfrei genannt, wenn die Verteilung der Knotengrade $P(k)$ einem Potenzgesetz $P(k) \sim k^{-\gamma}$ folgt (Barabás und Bonabeau, 2003). Der Wert für $\gamma$ wurde nach einer Methode von Newman ermittelt (Newman, 2005). Für diese Berechnung wird die Folge aller festgestellten Knotengrade $x = x_1, ..., x_i, ..., x_n$ für alle $n$ Knoten eines Graphen benötigt. Die Formel für den Faktor $\gamma$ des Potenzgesetzes lautet:

$$\gamma = 1 + n \left[ \sum_{i=1}^{n} \ln \frac{x_i}{x_{min}} \right]^{-1} , \tag{2.4}$$

wobei $x_{min}$ für den kleinsten Wert in $x$ steht.

## 2.6 Detektion von Ausreißern

Das Detektieren von Ausreißern erfolgte durch eine Methode von John W. Tukey (Tukey, 1977), die auch bei der Boxplot-Statistik zum Einsatz kommt. Ausreißer sind demnach Datenpunkte, die größer sind als das Dreifache des Interquartilsabstands (IQR). Der IQR ist definiert durch die Differenz des dritten und ersten Quantils:

$$IQR = Q_3 - Q_1. \tag{2.5}$$

## 2.7 SBML

Die *Systems Biology Markup Language* (SBML) (Hucka et al., 2003; Finney und Hucka, 2003) ist ein XML (Bray et al., 1998) basiertes Dateiformat zum Austausch biologischer Modelle. Das Format besteht aus zwei verschiedenen grundlegenden Elementen, den *Reactions* und den *Species*. Eine *Reaction* stellt beispielsweise eine biochemische Reaktion dar, deren Edukte und Produkte durch die *Species* repräsentiert werden. Ein weiteres wichtiges Element sind die *Modifier*, welche einer *Reaction* zugeordnet werden können. Diese *Modifier* sind Teil der biochemischen Reaktion, werden von dieser jedoch weder verbraucht noch erzeugt. Sie repräsentieren zum Beispiel Cofaktoren eines Enzyms oder das katalysierende Enzym selbst.

## 2.8 Modelle

Die in dieser Arbeit verwendeten Modelle stammen aus der *path2models*-Datenbank (Büchel et al., 2013), einem Zweig der *Biomodels*-Datenbank (Li et al., 2010). Es wurden der Datensatz der Gesamtgenom-Modelle einzelner Organismen (*whole genome metabolic models*) verwendet. Der verwendete Datensatz stammt aus der Version „r27“ der Datenbank und beinhaltet 2641 verschiedene Modelle. Die *path2models*-Datenbank enthält nicht manuell erzeugte Modelle, sondern Modelle, welche automatisiert von einem Softwarepaket erstellt

wurden. Somit sind alle Modelle nach den gleichen Regeln erstellt worden. Dadurch eignen diese Modelle sich sehr gut, um Vergleiche zwischen ihnen durchzuführen. Als Grundlage für diese Pipeline dienen alle vorhandenen Informationen zu einem Organismus aus der KEGG-Datenbank (Kanehisa et al., 2014). Diese Informationen wurden mit Einträgen aus der MetaCyc-Datenbank (Caspi et al., 2014) ergänzt, welche experimentell bestätigte Daten zu Stoffwechselwegen beinhaltet. Für eine eindeutige Zuordnung aller Enzyme werden diese im nächsten Schritt, sofern vorhanden, mit Uni-Prot-Identifikatoren (Consortium, 2008) und E.C.-Nummern (Webb, 1992) verknüpft. Metaboliten bekommen eine Zuordnung zu ChEBI-Identifikatoren (Hastings et al., 2013). Eine Zuordnung zu bestimmten Geweben oder Zellkompartimenten der einzelnen Reaktionen findet nicht statt, da hierfür nicht ausreichende Informationen vorhanden sind. Datenbanken für die Zuordnung eines Proteins zu einem bestimmten Zellkompartement existieren zwar, beispielsweise COMPARTMENT (Binder et al., 2014) oder Locate (Sprenger et al., 2008), jedoch ist eine solche Zuordnung nicht flächendeckend für alle 2641 Modelle möglich. Es wird jedoch zwischen einem intrazellulären und einem extrazellulären Bereich unterschieden. Die Einführung eines extrazellulären Bereiches ist nötig, da im nächsten Schritt zu jedem Modell ein minimales Wachstumsmedium hinzugefügt wird. Dieses besteht aus $\alpha$-D-Glukose, $\beta$-D-Glukose, Ammonium, Natrium, Kalium, Magnesium, Kalzium, Sulfat, Chlorat, Phosphat, Protonen, Wasser, Kohlenstoff, Dioxid und Sauerstoff. Wiederum mangels ausreichender Informationen über den Export von Metaboliten aus dem intra-zellulären Bereich, wurde pauschal für jeden Metaboliten eine solche Exportreaktion hinzugefügt. Der letzte Schritt ist die Einführung einer Biomasse-Reaktion, welche die 20 üblichen Aminosäuren, die nukleotiden Vorläufer der DNA und RNA, Glycogen und ATP in Biomasse umwandelt. Diese Reaktion ist bei allen Modellen gleich, sodass auch Bakterien oder Pflanzen Glycogen in ihre Biomasse einbauen. Ebenfalls nicht berücksichtigt wird, ob der Organismus, auf Grund der vorhandenen Informationen, alle Bausteine der Biomasse selbst produzieren kann.

## 2.9 Statistische Auswertung der topologischen Analysen

Die statistische Auswertung der durchgeführten topologischen Analysen sowie die Erzeugung der gezeigten Plots, erfolgte mit Hilfe des *R-Project* (R Core Team, 2015).

# Kapitel 3

# Ergebnisse und Diskussion

Im folgenden Kapitel werden die Ergebnisse der topologischen Analyse der 2641 Gesamtgenom-Modelle der *path2models*-Datenbank (Büchel et al., 2013) gezeigt. Für diese topologischen Analysen wurden die Modelle in bipartite Graphen, genauer in PN, umgewandelt, um die topologischen Eigenschaften sowohl der Metaboliten als auch der enzymatischen Reaktionen untersuchen zu können. Zu diesem Zweck wurde ein Kommandozeilenprogramm entwickelt, mit welchem die topologischen Analysen parallelisiert auf einem Computer-Cluster durchgeführt werden konnten. Die Resultate dieser Analysen werden als CSV-Dateien gespeichert und gleichzeitig in einer Datenbank hinterlegt. In der Datenbank wird auch die Struktur der PN gespeichert.

Im Folgendem wird zunächst das Kommandozeilenprogramm, die Datenbank und die Umwandlung der Modelle in PN beschrieben. Im Anschluss daran erfolgt eine Übersicht über die Ergebnisse der einzelnen topologischen Untersuchungen.

## 3.1 Petritopolis

Die topologischen Analysen der Gesamtgenom-Modelle erfolgte mit einem in Java implementierten Kommandozeilenprogramm namens Petritopolis. Die Aufgabe von Petritopolis ist es, die Modelle im SBML-Format einzulesen, zwischenzuspeichern, die topologischen Analysen durchzuführen und die Ergebnisse in CSV-Dateien und in einer Datenbank zu speichern. Petritopolis wurde entworfen, um diese Schritte auf einen Computer-Cluster parallelisiert durchführen zu können. Für die interne Repräsentation der Modelle als PN verwendet Petritopolis die selbe Klassenstruktur, wie die im ersten Teil dieser Arbeit vorgestellte Software MonaLisa. Die SBML-Dateien der Gesamtgenom-Modelle erfolgt mit Hilfe der jSBML-Bibliothek (Rodriguez et al., 2015). Bei der parallelen Berechnung bearbeitet jede Instanz von Petritopolis einen Teil der Modelle und speichert seine Teilergebnisse ab. Diese werden am Ende von Petritopolis zusammengesetzt und ausgewertet. Die Auswertung wird für jede topologische Eigenschaft in verschiedenen CSV-Dateien abgespeichert, um eine statistische Analyse mit R zu ermöglichen. Diese

Auswertung findet über alle Modelle und über alle Elemente dieser statt. Mit PETRITOPOLIS kann die Topologie der Modelle auch in einer Datenbank gespeichert werden. Während der Berechnung der einzelnen topologischen Eigenschaften werden in dieser die Werte für jedes einzelne Element des Modells gespeichert. So lassen sich später zum Beispiel Reaktionen oder Metaboliten mit einem hohem Knotengrad identifizieren. Das *Enhanced Entity-Relationship*-Diagramm (EER) der PETRITOPOLIS-Datenbank ist in Abbildung 3.1 gezeigt.

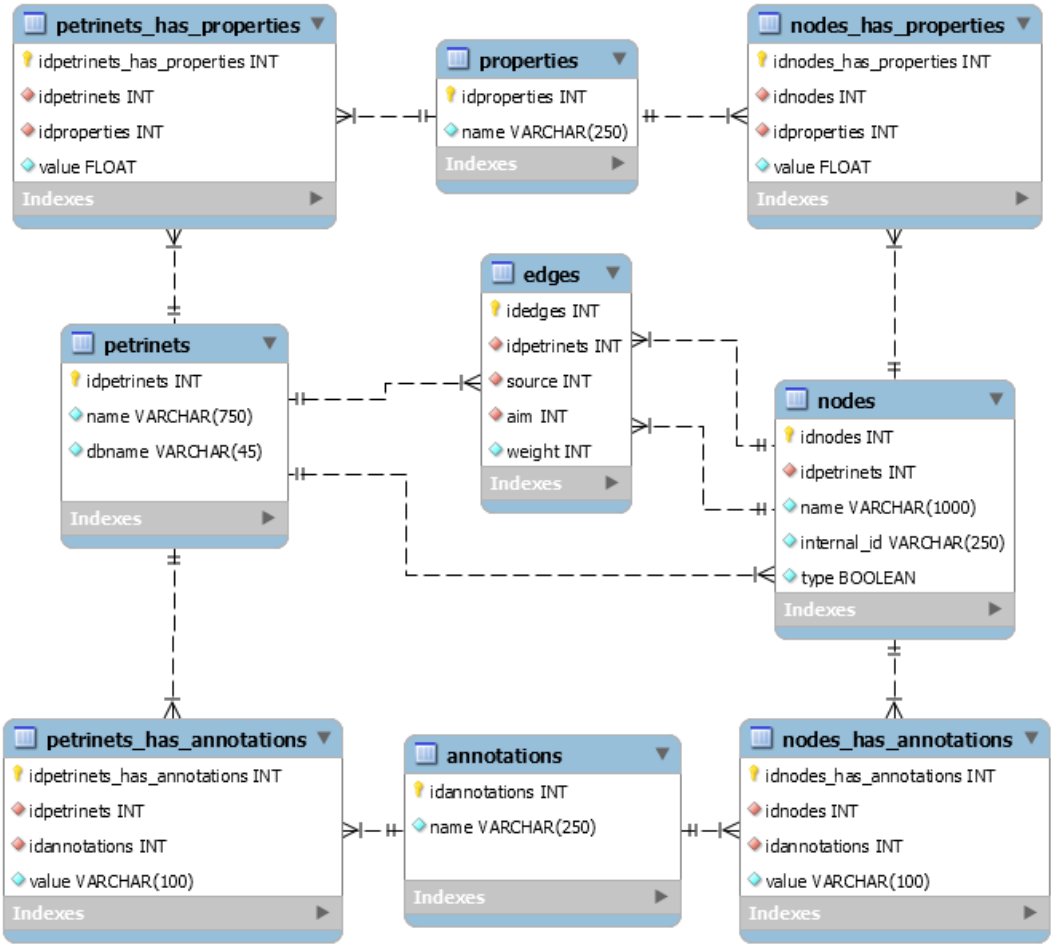


**Abbildung 3.1:** Die Abbildung zeigt das *Enhanced Entity-Relationship*-Diagramm der PETRITOPOLIS-Datenbank. Für jedes der Modelle wird zunächst ein Eintrag in der Tabelle *petrinets* angelegt, wobei der Name des Organismus und der Identifikator der *path2models*-Datenbank gespeichert wird. Die Knoten eines Modells, egal ob Reaktion oder Metabolit, werden in einer Tabelle gespeichert, in der deren Name und der Modell-interne Identifikator gespeichert wird. Das Feld *type* dient der Unterscheidung von Reaktionen und Metaboliten. Die Tabelle *properties* stellt alle vordefinierten Eigenschaften für ein Modell oder einen Knoten zur Verfügung. Dazu zählen Eigenschaften wie Knotengrad, Clusterkoeffizient oder die Anzahl von Reaktionen in einem Modell. Über die beiden Tabellen *petrinets_has_properties* und *nodes_has_properties* kann dann für die einzelnen Modelle oder Knoten der Wert einer bestimmten Eigenschaft hinterlegt werden. Nach dem gleichen Muster sind die Tabellen *annotations*, *petrinets_has_annotations* und *nodes_has_annotations* aufgebaut. Hier werden in den SBML-Dateien vorhandene Annotationen, wie zum Beispiel die genaue Taxonomie des Organismus oder die E.C.-Nummer einer Reaktion, hinterlegt.

Für jedes der Modelle wird zunächst ein Eintrag in der Tabelle *petrinets* angelegt, wobei der Name des Organismus und der Identifikator der *path2models*-Datenbank gespeichert wird. Die Knoten eines Modells, egal ob Reaktion oder Metabolit, werden in einer Tabelle gespeichert, in der deren Name und der Modell-interne Identifikator gespeichert wird. Das Feld *type* dient zur Unterscheidung von Reaktionen und Metaboliten. Die Tabelle *properties* stellt alle vordefinierten Eigenschaften für ein Modell oder einen Knoten zur Verfügung, wie zum Beispiel Knotengrad, Clusterkoeffizient oder Anzahl der Reaktionen in einem Modell. Über die beiden Tabellen *petrinets_has_properties* und *nodes_has_properties* kann dann für die einzelnen Modelle oder Knoten der Wert einer bestimmten Eigenschaft hinterlegt werden. Nach dem gleichen Muster sind die Tabellen *annotations*, *petrinets_has_annotations* und *nodes_has_annotations* aufgebaut. Hier werden in den SBML-Dateien vorhandene Annotationen, wie zum Beispiel die genaue Taxonomie des Organismus oder die E.C.-Nummer einer Reaktion (Webb, 1992), hinterlegt.

### 3.1.1 Konvertieren der Modelle in Petrinetze

Für die topologischen Analysen müssen die Modelle, vorliegend als SBML-Datei, in eine PN-Repräsentation übersetzt werden. Im ersten Schritt wird für jede *Reaction* eine entsprechende Transition angelegt und gegebenenfalls die reversible Transition erzeugt. Die von *path2models* erzeugten *Reactions* repräsentieren nicht ein spezielles Enzym sondern eine spezifische biochemische Reaktion. Alle Enzyme, die diese Reaktion katalysieren, werden als *Modifier* für diese hinterlegt. Dazu gehören zum Beispiel verschiedene Isoformen desselben Enzyms oder verschiedene Enzyme, die diese Reaktion in verschiedenen Geweben katalysieren. Aus diesem Grund muss beim Erzeugen der Plätze berücksichtigt werden, ob es sich bei der *Species* um einen Metaboliten oder einen *Modifier* handelt, welcher ignoriert werden muss. Dieses Vorgehen wurde gewählt, da in den Modellen nicht zwischen verschiedenen Geweben oder Zellkompartimenten eines Organismus unterschieden wird. Somit wird von einem einzigen großen Metaboliten-Pool ausgegangen. Daher muss auch von nur einem „Reaktions-Pool“ ausgegangen werden, in dem jede enzymatische Reaktion des Organismus nur einmal vorkommt. Ein mehrmaliges Modellieren der Reaktion für jedes der Enzyme, ohne Auftrennung in Metaboliten-Pools, würde zu einer Verfälschung der topologischen Eigenschaften führen. Bei der Erstellung der Modelle wurde in jedem Modell eine Reaktion zur Produktion von Biomasse eingeführt, auch dann, wenn dessen Bausteine nicht komplett vom Organismus hergestellt werden können. Deshalb wird diese Reaktion aus dem PN entfernt. Ebenfalls entfernt werden die künstlich eingeführten Exportreaktionen der Metaboliten sowie deren extrazellulären Gegenstücke. Als letzter Schritt werden mögliche isolierte Bereiche des Modells gelöscht, sodass das endgültige PN eine einzige Zusammenhangskomponente bildet.

Durch die Umwandlung der als monopartiten Graphen, also ein Graph mit nur einer Knotenmenge statt mit zwei disjunkten Knotenmengen, modellierten Systeme in PN, werden die topologischen Eigenschaften der einzelnen Metaboliten teilweise verändert. So

beispielsweise bei einer Reaktion vom Typ $A+B \rightarrow C$. Bei der Modellierung monopartiter Graphen wird die Reaktion als Kante zwischen den Edukten und Produkten eingefügt, Metabolit $C$ hat also einen Knotengrad von 2. In einem PN werden diese Kanten durch einen zusätzlichen Knoten ersetzt, und von diesem führt nur noch eine Kante zu Metabolit $C$, dessen Knotengrad beträgt also nur noch $k = 1$. Dies bildet die eigentliche Struktur der Reaktionen besser ab, da durch die beschriebene Reaktion Metabolit $C$ ja nicht an zwei Reaktionen teilnimmt.

## 3.2 Verhältnis der Anzahl von Reaktionen und Metaboliten

Vor der Untersuchung der topologischen Eigenschaften der Modelle, wurde die Frage untersucht, wie sich das Verhältnis der Anzahl von Reaktionen zu der Anzahl von Metaboliten in den Modellen verhält. Das Ergebnis ist in Abbildung 3.2 dargestellt.

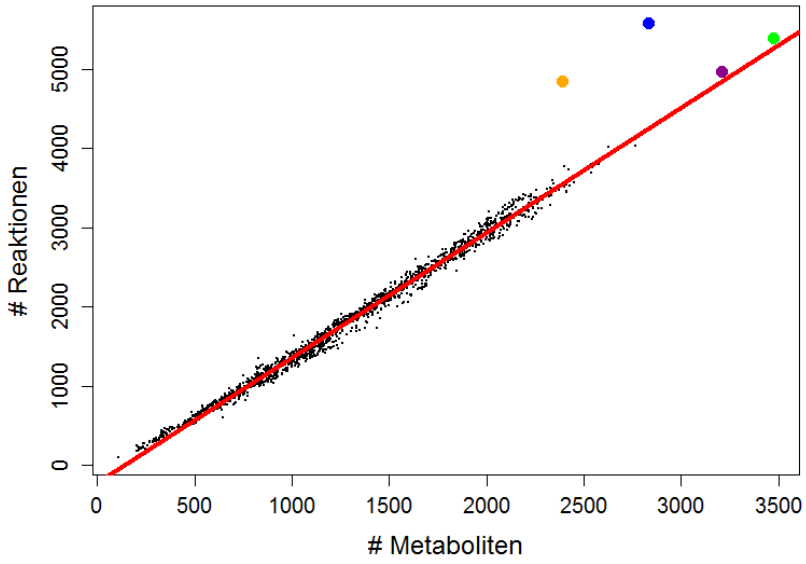


**Abbildung 3.2:** Jeder Punkt im Plot steht für ein einzelnes Modell. Auf der x-Achse ist die Anzahl der Metabolien aufgetragen, auf der y-Achse die Anzahl der Reaktionen. Die eingepasste rote Linie hat eine Steigung von $1.574 \pm 0.003$, im Durchschnitt besitzt ein Modell also 1.57 mal mehr Reaktionen als Metabolite. Das größte Modell (grün) ist das Gesamtgenom-Model von *Salmonella enterica subsp. enterica serovar Paratyphi B* mit 5392 Reaktionen und 3473 Metaboliten. Das Modell des menschlichen Metabolismus ist lila eingefärbt und umfasst 4964 Reaktionen und 3209 Metaboliten. Der blaue und orangene Punkt repräsentiert jeweils die Modelle von *Drosophila melanogaster* und seines Endosymbionten *Wolbachia pipientis wMel*. Das Verhältnis von Reaktionen und Metaboliten ist in diesen Modellen etwas höher und beträgt 1.97 beziehungsweise 2.03.

Jeder Punkt in diesem Plot stellt eines der 2641 Modelle dar. Auf der x-Achse ist die Anzahl der Metaboliten aufgetragen, auf der y-Achse die Anzahl der Reaktionen. Die Größe der Modelle reicht von Modellen mit nur sehr wenigen Reaktionen bis zu dem Model des Metabolismus von *Salmonella enterica subsp. enterica serovar Paratyphi B*, gegeben durch einen grünen Punkt, mit 5392 Reaktionen und 3473 Metaboliten. Das Modell des

menschlichen Metabolismus wird durch einen lila Punkt repräsentiert und umfasst 4964 Reaktionen und 3209 Metaboliten. Die Anzahl der Reaktionen eines Modelles $n_r$ korreliert stark mit der Anzahl der Metaboliten in diesem $n_m$ und ist annähernd $n_r = 1.57 \times n_m$. Im Durchschnitt hat ein Gesamtgenom-Modell 1.57 mal mehr Reaktionen als Metaboliten. Ausnahmen bilden hier die Modelle von *Drosophila melanogaster*, blauer Punkt, und seines Endosymbionten, dem gramnegativen Bakterium *Wolbachia pipientis wMel* (Olsen et al., 2001), orangener Punkt. In diesen Modellen gibt es 1.97 mal, respektive 2.03 mal, mehr Reaktionen als Metaboliten.

Das Verhältnis der Anzahl an Reaktionen und Metaboliten ist über alle Modelle sehr ähnlich. Die Modelle umfassen Organismen aus allen Reichen, diese Eigenschaft ist also unabhängig von diesem. Bei steigender Anzahl an verschiedenen biochemischen Reaktionen eines Organismus steigt also auch die Diversität der von diesem Organismus umgesetzten und produzierten chemischen Verbindungen. Es werden also nicht nur neue Reaktionen zwischen den vorhandenen Metaboliten hinzugefügt. Auf das erhöhte Verhältnis von Reaktionen zu Metaboliten bei den beiden Modellen von *Drosophila melanogaster* und seines Endosymbionten *Wolbachia pipientis wMel* wird im weiteren Text noch näher eingegangen, da hierzu weitere topologische Eigenschaften benötigt werden.

## 3.3 Knotengradverteilung der Metaboliten und Reaktionen

### 3.3.1 Knotengrad der Metaboliten

Der Knotengrad eines Metaboliten $k_n$ gibt an, an wie vielen Reaktionen dieser beteiligt ist. Der eingehende Knotengrad $k_n^i$ steht dabei für die Anzahl an Reaktionen, die diesen Metaboliten erzeugen, der ausgehende Knotengrad $k_n^o$ für die Anzahl an Reaktionen, die diesen Metaboliten konsumieren. Abbildung 3.3 zeigt die Wahrscheinlichkeitsverteilung des Knotengrades über alle Modelle der *path2models*-Datenbank. Beide Achsen sind logarithmiert, auf der x-Achse ist der Knotengrad $k$ aufgetragen, auf der y-Achse die Wahrscheinlichkeit $p(k)$, mit der ein Metabolit diesen Knotengrad besitzt. Ein Punkt ist rot, blau oder orange eingefärbt, wenn allein $H^+$, $H_2O$ beziehungsweise eine Mischung aus beiden Stoffen zu diesem Knotengrad beitragen. Der niedrigste Knotengrad, den ein Metabolit haben kann, ist $k = 1$ und tritt mit einer Wahrscheinlichkeit von $p(k = 1) = 0.2$ auf. Beispiele hierfür sind $HCO_3^-$ oder Aminosäuren wie D-Glutamat. Der Knotengrad mit der höchsten Wahrscheinlichkeit ist $p(k = 2) = 0.42$. Metaboliten mit diesem Knotengrad werden entweder von einer einzigen Reaktion erzeugt oder verbraucht, wie zum Beispiel Desoxycytidin-Diphosphat oder Arachidonsäure. Dem gegenüber stehen sehr hohe Knotengrade. Der höchste gefundene Knotengrad ist $k = 3012$ und tritt für das Wasserstof-Ion $H^+$ im Modell von *Drosophila melanogaster* auf. Wasserstoff-Ionen sind in mehr als 99% der Modelle mit einem sehr hohem Knotengrad vorhanden. Beispiele für weitere Metaboliten mit einen hohem Knotengrad sind $H_2O$, $NADP^+$ (auch in seiner reduzierten Form NADPH), $O_2$, ATP und $CO_2$. Fast alle hohen Knotengrade größer als $k = 500$ können

$H^+$ oder $H_2O$ zugeschrieben werden. Der durchschnittliche Knotengrad eines Metaboliten beträgt 6.42.

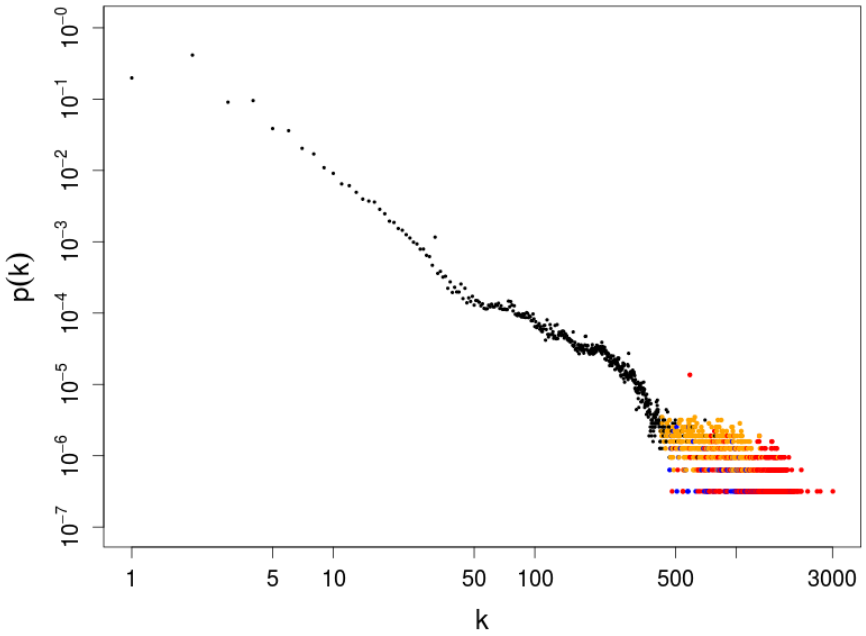


**Abbildung 3.3:** Der log-log-Plot zeigt die Verteilung der Knotengrade der Metaboliten in allen Modellen. Ein Punkt gibt die Wahrscheinlichkeit $p(k)$ an, mit der ein Metabolit den Kontengrad $k$ hat. Die Wahrscheinlichkeit $p(k)$ fällt für ansteigende Knotengrade, es treten jedoch Knotengrade bis zu $k = 3012$ in den Modellen auf. Ein Punkt ist rot, blau oder orange eingefärbt, wenn allein $H^+$, $H_2O$ beziehungsweise eine Mischung aus beiden Stoffen zu diesem Knotengrad beitragen. Das gewichtete arithmetische Mittel der Verteilung beträgt 6.42.

Die entsprechenden Abbildungen für den eingehenden Knotengrad $k^i$ und den ausgehenden Knotengrad $k^o$ für die Metaboliten sind im Anhang als Abbildung 5.1 beziehungsweise 5.2 zu finden.

Aus der Literatur ist bekannt, dass die Wahrscheinlichkeitsverteilung des Knotengrades für Metaboliten in metabolischen Netzwerken skalenfrei ist (Jeong et al., 2000). Diese Eigenschaft liegt auch in der vorliegenden Verteilung vor. Für die Bestimmung des $\gamma$-Wertes des Potenzgesetzes, welcher einer skalenfreien Verteilung zugrunde liegt, sind jedoch einige Einschränkungen zu beachten. Bei einer skalenfreien Verteilung ist die Wahrscheinlichkeit für sehr hohe Knotengrade niemals 0. Da durch die Größe des Netzwerkes aber der maximale Knotengrad eine obere Grenze hat, ist dies nicht beweisbar. Daher ist das Potenzgesetz nur auf einen begrenzten Bereich der Verteilung anwendbar (Steuer und López, 2007).

Für die Bestimmung dieses Bereiches betrachten wir die Kumulante der Wahrscheinlichkeitsverteilung, welche in Abbildung 3.4 gezeigt wird. Diese ist eine monoton fallende Funktion und gibt die Wahrscheinlichkeit $p_c(k)$ an, dass ein beliebiger Knoten einen Knotengrad höher als $k$ hat. Beide Achsen sind logarithmiert. Die Wahrscheinlichkeit für einen

Knotengrad höher als $k$ nimmt für Knotengrade bis ungefähr $k = 1500$ langsam ab. Für Knotengrade $k > 1500$ fällt die Wahrscheinlichkeit drastisch ab. Wird nun der $\gamma$-Wert der Wahrscheinlichkeitsverteilung der Knotengrade im Bereich $0 \geq k \leq 1500$ bestimmt, so ergibt sich $\gamma = 2.010743 \pm 0.0006$ als Faktor im Potenzgesetz.

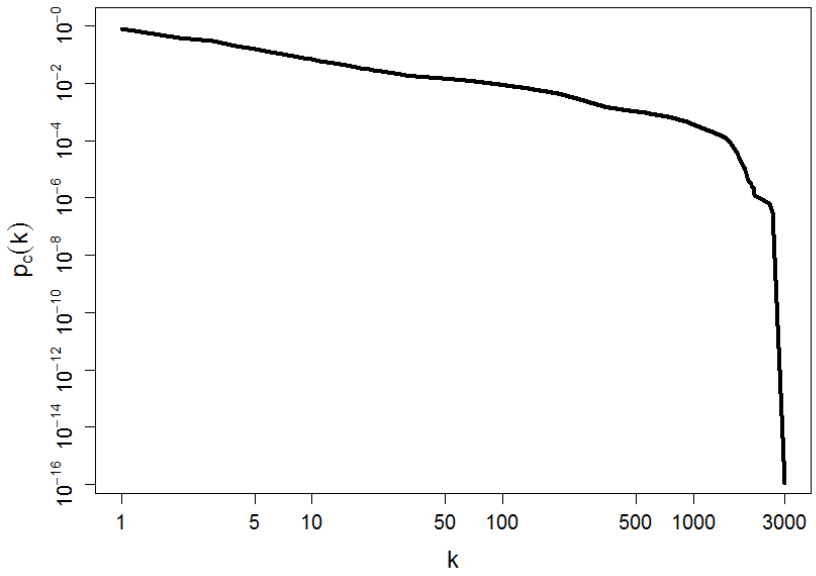


**Abbildung 3.4:** Die Abbildung zeigt die log-log-Darstellung der Kumulante der Wahrscheinlichkeitsverteilung des Knotengrades aller Metaboliten. Auf der x-Achse ist der Knotengrad $k$ aufgetragen. Die y-Achse gibt die Wahrscheinlichkeit $p_c(k)$ an, dass ein beliebiger Knoten einen Knotengrad höher als $k$ hat. Bis zu einem Kontengrad von ungefähr $k = 1500$ fällt die Kumulante gleichmäßig ab. Für Werte $k > 1500$ fällt die Kumulante stark ab.

Ein Metabolit hat ein Knotengrad $k$ mit der in Abbildung 3.3 gezeigten Wahrscheinlichkeit. Für einen Knotengrad $k$ gibt es mehrere Kombinationen aus $k^i$ und $k^o$, die genau $k$ ergeben. Abbildung 3.5 zeigt für die Knotengrade $1 \geq k \leq 4$ alle möglichen Kombinationen aus $k^i$ und $k^o$, um einen Knotengrad $k$ zu erreichen. Der Grad der roten Ausfüllung des Kreises einer dieser Kombinationen korreliert mit der relativen Häufigkeit dieser, innerhalb des selben Knotengrades. Zu beobachten ist, dass ein möglichst ausgeglichenes Verhältnis von $k^i$ und $k^o$ Kanten bevorzugt wird. Je größer die Differenz zwischen eingehenden und ausgehenden Kanten ist, desto seltener tritt dieser Fall auf. Für gerade Knotengrade ist es am wahrscheinlichsten, dass ein Metabolit genau so viele eingehende wie ausgehende Kanten hat. Für Metaboliten gilt in den meisten Fällen, dass es annähernd so viele Reaktionen gibt, die diese verbrauchen, wie Reaktionen die diese erzeugen. Wird für jeden Metaboliten der Wert $\Delta k = k^i - k^o$ gebildet und das gewichtete arithmetische Mittel über die Wahrscheinlichkeitsverteilung von $\Delta k$ gebildet, so ergibt sich ein Wert von $-0.23$. Die dazugehörige Abbildung ist im Anhang als Abbildung 5.3 zu finden. Ein negativer Wert für $\Delta k$ bedeutet, dass ein Metabolit von mehr Reaktionen verbraucht als erzeugt wird. Trotz der Tendenz zu einem ausgeglichenen Verhältnis von $k^i$ zu $k^o$, werden Metaboliten in der Gesamtheit von mehr Reaktionen verbraucht, als erzeugt. Der Wertebereich von

$\Delta k$ reicht von $-490$ bis 166. Metaboliten mit einem sehr hohem negativen $\Delta k$ sind beispielsweise $H^+$ oder $H_2O$. Sehr hohe positive Werte für $\Delta k$ weisen Metaboliten wie zum Beispiel $NADP^+$ oder oder $CO_2$ auf.

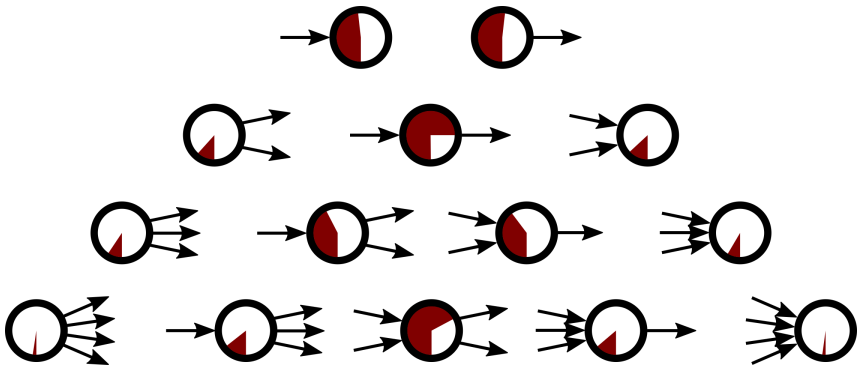

**Abbildung 3.5:** Für jeden Knotengrad $k$ eines Metaboliten im Bereich $1 \geq k \leq 4$ werden alle möglichen Kombinationen aus $k^i$ und $k^o$ aufgezeigt. Der Grad der roten Ausfüllung des Kreises einer dieser Kombination korreliert mit der relativen Häufigkeit dieser, innerhalb des selben Knotengrades. Über alle Knotengrade hinweg ist zu beobachten, dass ein möglichst ausgeglichenes Verhältnis von $k^i$ und $k^o$ bevorzugt wird.

Für jedes Modell kann der durchschnittliche Knotengrad aller Metaboliten ermittelt werden. In Abbildung 3.6 ist dieser Wert, auf der y-Achse, im Verhältnis zur Anzahl der Metaboliten des Modells, x-Achse, gesetzt. Kleinere Modelle haben auch einen niedrigeren durchschnittlichen Knotengrad ihrer Metaboliten, der minimale Wert ist 3.25 für das Modell von *Candidatus Tremblaya princeps PCVAL*, einem nicht klassifizierten Betaproteobakterium. Der Wert steigt bis auf 7.46 für größere Modelle an. Einen deutlich höheren Wert haben die beiden Modelle von *Drosophila melanogaster*, blau hervorgehoben, und seines Endosymbionten *Wolbachia pipientis wMel*, orange hervorgehoben. Hier liegt er bei 9.21 beziehungsweise 9.51.

Die Wahrscheinlichkeitsverteilung des Knotengrades aller Metaboliten ist wie gezeigt skalenfrei (siehe Abbildung 3.3). Diese Eigenschaft wurde für metabolische Modelle schon oft gezeigt und diskutiert (Jeong et al., 2000; Tanaka, 2005; Wagner und Fell, 2001). Der $\gamma$-Wert des zugrundeliegenden Potenzgesetzes beträgt $\gamma = 2.010743 \pm 0.0006$ und ist somit im Bereich, welcher für metabolische Modelle typisch ist (Albert und Barabási, 2002). Somit ist auch durch die Umwandlung der Modelle in bipartite Graphen diese Eigenschaft nicht verloren gegangen.

Auch das in Abbildung 3.6 gezeigte Verhalten, dass der durchschnittliche Knotengrad der Metaboliten in einem Modell steigt, wenn die Anzahl der Metaboliten in einem Modell steigt, ist aus der Literatur bekannt (Albert und Barabási, 2002). Dort wird es dem Phänomen des beschleunigten Wachstums zugeschrieben, bei dem die Anzahl der Kanten in einem Modell stärker zunimmt als die Anzahl der Knoten.

Das gewichtete arithmetische Mittel über die Verteilung von $\Delta k = k^i - k^o$ von $-0.23$

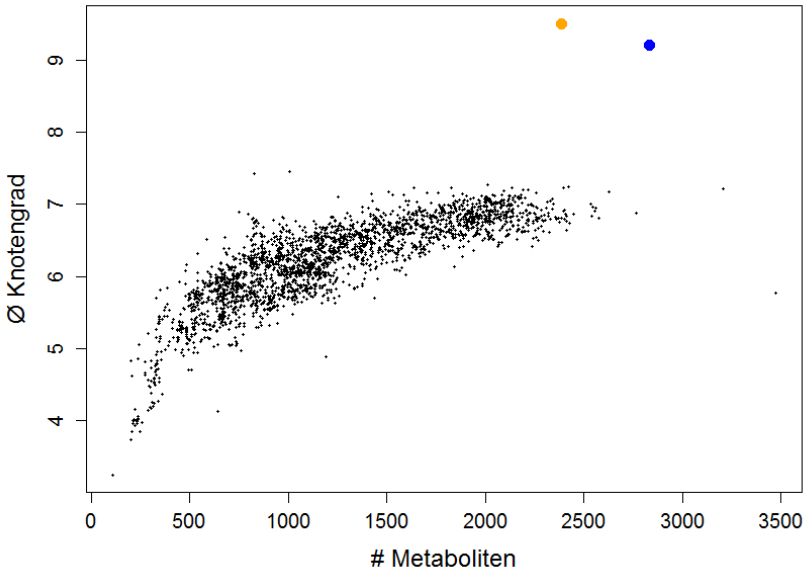


**Abbildung 3.6:** Jeder Punkt des Plots steht für eins der 2641 Modelle. Auf der x-Achse ist die Anzahl der Metaboliten in diesem Modell aufgetragen. Die y-Achse gibt den durchschnittlichen Knotengrad aller Metaboliten in diesem Modell an. Der blaue Punkt repräsentiert das Modell von *Drosophila melanogaster*, der orangene Punkt das seines Endosymbionten *Wolbachia pipientis wMel*. Mit steigender Anzahl an Metaboliten in einem Modell, steigt auch deren durchschnittlicher Knotengrad.

sagt aus, dass die Metaboliten in den Modellen öfters als Edukt einer Reaktion auftreten anstatt als Produkt. Das heißt, dass die Diversität der Metaboliten im Modell leicht abnimmt, da durch Reaktionen die Menge verschiedener Metaboliten reduziert wird. Ein Organismus sichert durch seinen Metabolismus nicht nur seine Energieversorgung, sondern muss auch Biomasse akkumulieren, welche aus vielen einzelnen Metaboliten zusammengesetzt werden muss, um zum Beispiel neue Zellmembranen zu erzeugen. Die Reaktionen zur Erzeugung von Biomasse ist zwar aus den Reaktionen entfernt, die Erzeugung von deren Bausteine ist aber weiterhin in den Modellen enthalten.

### 3.3.2 Knotengrad der Reaktionen

Der Knotengrad einer Reaktion $k_n$ gibt an, wie viele Produkte und Edukte an dieser Reaktion beteiligt sind. Der eingehende Knotengrad $k_n^i$ steht dabei für die Anzahl an konsumierten Metaboliten der Reaktion, der ausgehende Knotengrad $k_n^o$ für die Anzahl an produzierten Metaboliten. Abbildung 3.7 zeigt die Wahrscheinlichkeitsverteilung des Knotengrades über alle Modelle der *path2models*-Datenbank. Auf der x-Achse ist der Knotengrad $k$ aufgetragen, auf der y-Achse die Wahrscheinlichkeit $p(k)$, mit der eine Reaktion diesen Knotengrad besitzt. Knotengrade von $k < 2$ existieren für keine Reaktion, der maximale Knotengrad ist $k = 13$, wobei es sich hierbei ausschließlich um die Reaktionen der Ferribaction- oder der Yersiniabaction-Synthese handelt. Mit $p(k = 5) = 0.34$ und $p(k = 4) = 0.29$ sind diese beiden Knotengrade am wahrscheinlichsten für eine Reaktion.

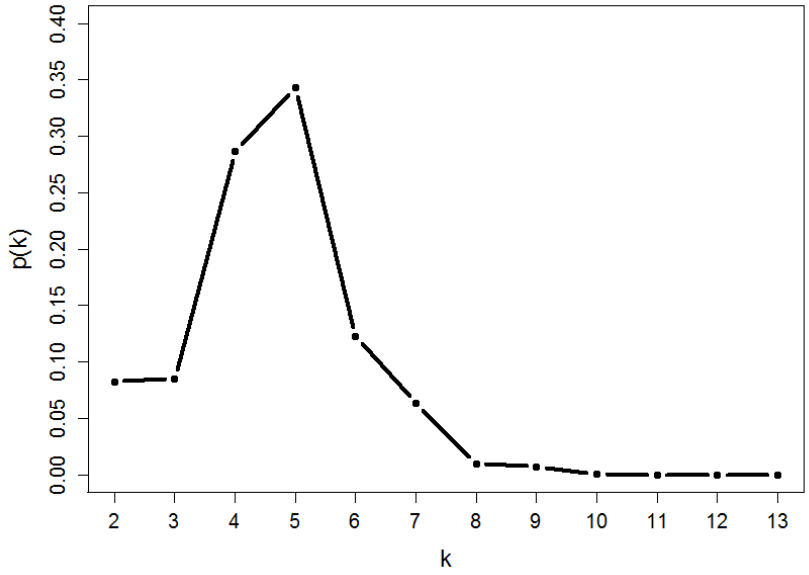


**Abbildung 3.7:** Der Plot zeigt die Verteilung der Knotengrade der Reaktionen in allen Modellen. Ein Punkt gibt die Wahrscheinlichkeit $p(k)$ an, mit der eine Reaktion den Kontengrad $k$ hat. Der höchste Wert der Verteilung ist $p(k = 5) = 0.34$, gefolgt von $p(k = 4) = 0.29$. Ein Knotengrad von $k < 2$ tritt für keine Reaktion auf. Es existieren jedoch Reaktionen bis zu einem Knotengrad von $k = 13$, mit einem Wert von $p(k = 13) = 2.27e - 5$. Dabei handelt es sich ausschließlich um die Reaktionen der Ferribaction- oder der Yersiniabaction-Synthese. Das gewichtete arithmetische Mittel der Verteilung beträgt 4.6.

Die entsprechenden Abbildungen für den eingehenden Knotengrad $k^i$ und den ausgehenden Knotengrad $k^o$ für Reaktionen sind im Anhang als Abbildung 5.4 beziehungsweise 5.5 zu finden.

Der Knotengrad $k$ für eine Reaktion kann aus verschiedenen Kombinationen von $k^i$ und $k^o$ gebildet werden. In Abbildung 3.8 sind alle Kombinationen von $k^i$ und $k^o$ für den Bereich $2 \geq k \leq 5$ aufgezeigt.

Die Größe der roten Ausfüllung des Kreises in einer Kombination steht in Korrelation zu dessen relativer Häufigkeit innerhalb eines Knotengrades. Kombinationen, die für keine Reaktion in allen Modellen auftreten, werden der Übersicht halber nicht gezeigt. Zu solchen gehören zum Beispiel alle Kombinationen, bei denen es nur eingehende oder nur ausgehende Kanten gibt, da keine Reaktion aus dem Nichts Stoffe erzeugt, oder diese einfach vernichtet. Wie bei den Metaboliten ist auch hier zu beobachten, dass ein möglichst ausgeglichenes Verhältnis von $k^i$ und $k^o$ Kanten bevorzugt wird. Im Unterschied zu dem Metaboliten gibt es jedoch eine Tendenz zu Kombinationen mit einem höheren $k^i$ als $k^o$. Kombinationen, bei denen der Unterschied $\Delta k = k^i - k^o$ größer als $\pm 1$ ist, sind sehr selten, Werte von $\Delta k = \pm 2$ treten so gut wie gar nicht auf, der gesamte Wertebereich ist $-4 >= \Delta k <= 6$. Ein positiver Wert für $\Delta k$ bedeutet, dass eine Reaktion mehr Metaboliten konsumiert als produziert. Die Wahrscheinlichkeitsverteilung der Werte für $\Delta k$ ist

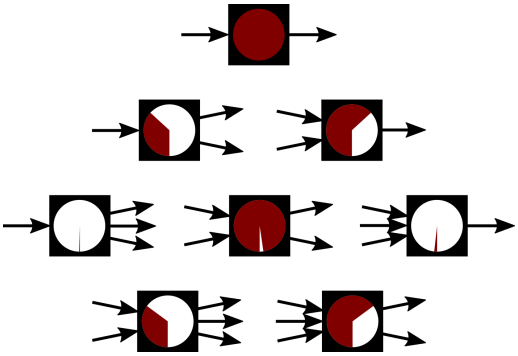

**Abbildung 3.8:** Für jeden Knotengrad $k$ einer Reaktion im Bereich $2 \geq k \leq 5$ werden alle möglichen Kombinationen aus $k^i$ und $k^o$ aufgezeigt. Der Grad der roten Ausfüllung des Quadrats einer dieser Kombinationen korreliert mit der relativen Häufigkeit dieser, innerhalb des selben Knotengrades. Kombinationen, die nicht auftreten, werden der Übersicht halber nicht aufgeführt. Bei keinem gezeigten Knotengrad gibt es Kombinationen, bei der es nur eingehende oder nur ausgehende Kanten gibt. Auch bei den Reaktionen wird ein ausgeglichenes Verhältnis von $k^i$ und $k^o$ bevorzugt. Allerdings gibt es eine Tendenz zu Kombinationen mit einem höheren $k^i$ als $k^o$.

im Anhang als Abbildung 5.6 zu finden. Das gewichtete arithmetische Mittel dieser Verteilung beträgt $0,17$. Reaktionen konsumieren in ihrer Gesamtheit also mehr Metaboliten als sie produzieren. Als Beispiel für Reaktionen mit einem hohem positiven $\Delta k$ sind die Phosphomethylpyrimidin- oder die Ferribactin-Synthese zu nennen. Hohe negative Werte für $\Delta k$ weisen beispielsweise die reverse Isopentenyl-Adenosin-A37-tRNA-Methylthiolase oder Thioglucoside-Glucohydrolase auf.

Auch für die Reaktionen kann deren durchschnittlicher Knotengrad in einem Modell ermittelt werden. In Abbildung 3.9 ist dieser Wert, auf der y-Achse im Verhältnis zur Anzahl der Reaktionen des Modells, x-Achse, gesetzt.

Der durchschnittliche Knotengrad einer Reaktion ist nur für sehr kleine Modelle, mit weniger als 500 Reaktionen, niedriger. Für die restlichen Modelle bewegt er sich zwischen 4.4 und 4.8. Der minimale Wert ist 3.65 für das Modell des *Hydrogenobaculum sp. 3684* und beträgt maximal 4.8 für das Modell von *Natronomonas pharaonis (strain DSM 2160 / ATCC 35678)*. Die beiden Modelle von *Drosophila melanogaster*, blau hervorgehoben, und seines Endosymbionten *Wolbachia pipientis wMel*, orange hervorgehoben, haben jedoch keinen erhöhten durchschnittlichen Knotengrad ihrer Reaktionen.

Die Wahrscheinlichkeitsverteilung der Knotengrade aller Reaktionen (siehe Abbildung 3.7) weicht von der erwarteten Verteilung ab. Typische biochemische Reaktionen sind Reaktionen erster Ordnung, also $A \rightarrow B$, oder Reaktionen zweiter Ordnung, also $A + B \rightarrow C$ oder $2A \rightarrow B$. Das Maximum der Verteilung wäre demnach um die Knotengrade 2 und 3 herum zu erwarten. Dies ist jedoch nicht der Fall, vielmehr liegt das Maximum um die Knotengrade 4 und 5 herum. Der Grund hierfür kann die Berücksichtigung der sogenannten Sekundärmetaboliten bei der Modellierung sein. Mit Sekundärmetaboliten sind Metaboliten wie beispielsweise ATP, NADPH oder $H^+$ gemeint, die an vielen biochemischen

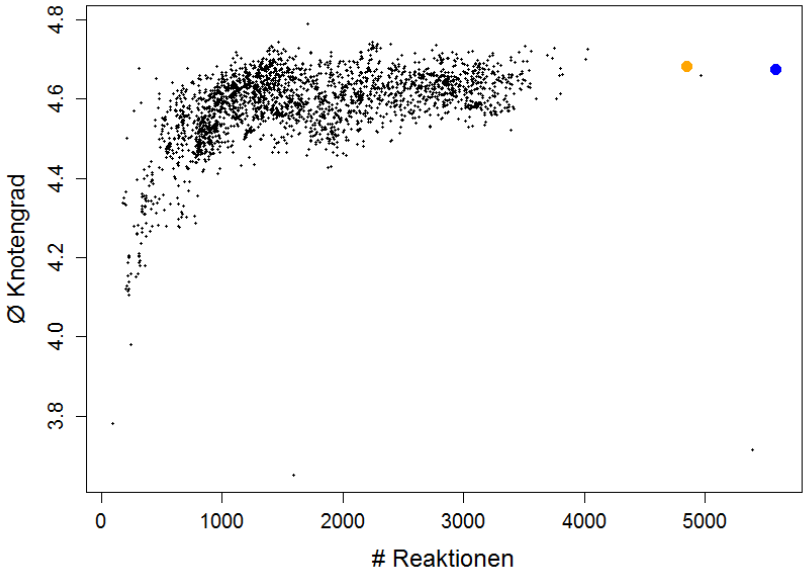


**Abbildung 3.9:** Jeder Punkt des Plots steht für eins der 2641 Modelle. Auf der x-Achse ist die Anzahl der Reaktionen in diesem Modell aufgetragen. Die y-Achse gibt den durchschnittlichen Knotengrad aller Reaktionen in diesem Modell an. Der blaue Punkt repräsentiert das Modell von *Drosophila melanogaster*, der orangene Punkt das seines Endosymbionten *Wolbachia pipientis wMel*. Der durchschnittliche Knotengrad einer Reaktion ist nur für sehr kleine Modelle (Anzahl der Reaktionen $< 500$) niedriger. Für die restlichen Modelle bewegt er sich zwischen 4.4 und 4.8.

Reaktionen teilhaben, und dabei Funktionen wie zum Beispiel das Bereitstellen von Protonen oder Energie haben. Nimmt ein Sekundärmetabolit an einer Reaktion teil, so wird dieser im Laufe der Reaktion meist in einen anderen Metaboliten umgewandelt. Dadurch erhöht sich der Knotengrad einer Reaktion um zwei, was die Verschiebung des Maximums erklären könnte. Um diese These zu überprüfen, müssen die Sekundärmetaboliten aus den Modellen entfernt werden. Der Begriff der Sekundärmetaboliten ist in Unterschiedlichen Bereichen der Biologie unterschiedlich definiert. In dieser Arbeit zählen zu den Sekundärmetaboliten all jene Metaboliten eines Modelles, die einen überdurchschnittlich hohen Knotengrad aufweisen, sie stellen also Ausreißer in der Verteilung dar. Deshalb wurden mit einer Methode zur Detektion von Ausreißern diese aus den Modellen entfernt. In Abbildung 3.10 ist die Wahrscheinlichkeitsverteilung der Knotengrade der Reaktionen nach Entfernen der Sekundärmetaboliten in blau gezeigt.

In schwarz ist die ursprüngliche Verteilung eingezeichnet. Es ist deutlich zu erkennen, dass das Maximum der Verteilung nun bei $k = 2$ liegt und auch die Wahrscheinlichkeit für $k = 3$ angestiegen ist. Diese Annahme wird auch von den Verteilungen der Kombinationen von $k^i$ und $k^o$ der Reaktionen (siehe Abbildung 3.8) unterstützt. Für $k = 4$ und $k = 5$ treten fast ausschließlich solche Kombinationen auf, die durch das Hinzufügen von Sekundärmetaboliten zu Reaktionen erster oder zweiter Ordnung gebildet werden. Diese Kombinationen sind auch solche mit einem sehr ausgeglichenen Verhältnis von $k^i$ und $k^o$. Ein weiterer Grund für die starke Tendenz zu solchen Kombinationen ist, dass chemische

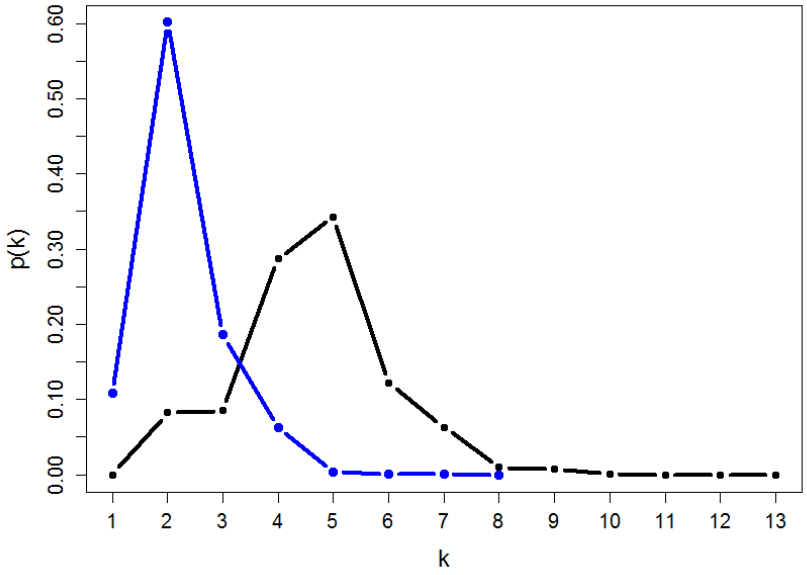


**Abbildung 3.10:** Der Plot zeigt die Verteilung der Knotengrade der Reaktionen in allen Modellen vor (schwarz) und nach (blau) dem Entfernen der Sekundärmetaboliten. Ein Punkt gibt die Wahrscheinlichkeit $p(k)$ an, mit der eine Reaktion den Kontengrad $k$ hat. Es ist deutlich erkennbar, dass sich das Maximum der Verteilung nach Entfernen der Sekundärmetaboliten von $k = 5$ auf $k = 3$ verschiebt.

Verbindungen, um eine biochemische Bedeutung nicht zu verlieren, nicht in ihre einzelnen Atome zerlegt werden können. So ist die Auftrennung eines Metaboliten durch eine Reaktion in drei unterschiedliche neue Substanzen nur möglich, wenn dieser aus einer entsprechend hohen Anzahl von Atomen besteht, die eine Auftrennung in drei biochemisch relevante Metaboliten zulässt. Andersherum können zwar beliebige Metaboliten zu einem großem Metaboliten zusammensetzt werden, aber auch für die Größe eines Metaboliten gibt es chemische und physikalische Grenzen.

Zudem zeigt sich, dass die Wahrscheinlichkeitsverteilung der Knotengrade der Reaktionen, im Gegensatz zu der der Metaboliten, nicht skalenfrei ist. Der maximale Knotengrad der Reaktionen ist um ein vielfaches geringer als der maximale Knotengrad eines Metaboliten, was durch physikalische und chemische Bedingungen gegeben ist. Ein Enzym, welches Bindestellen für mehrere tausend Metaboliten aufweisen würde, müsste eine enorme Größe haben, was auch die räumliche Zusammenführung der einzelnen Metaboliten unmöglich machen würde. In einer Arbeit von Wagner und Fell (2001) wird zwar aufgezeigt, dass die Verteilung des Knotengrades für einen sogenannten *Reaction Graph* ebenfalls skalenfrei ist, jedoch ist ein solcher nicht mit einem PN-Modell vergleichbar. In einem *Reaction Graph* sind die Reaktionen als Knoten dargestellt, und zwei Reaktionen durch eine ungerichtete Kante verbunden, falls beide Reaktionen mindestens einen Metaboliten gemeinsam haben, entweder als Edukt oder als Produkt. Teilen zwei Reaktionen mehr als einen Metaboliten, existiert trotzdem nur eine Kante. Zudem wird durch ungerichtete Kanten die Tatsache ignoriert, dass nicht alle Reaktionen reversibel sind. Untersucht wurde in der Arbeit von

Wagner und Fell (2001) der Metabolismus von *E. Coli*, der *Reaction Graph* weist für diesen Organismus einen maximalen Knotengrad von $k = 128$ auf, also einen deutlich höheren Wert als in den hier untersuchten PN-Modellen.

Nach ausgiebiger Betrachtung des Knotengrades von Metaboliten und Reaktionen ist nun auch eine Erklärung des erhöhten Verhältnisses von Metaboliten und Reaktionen in den Modellen von *Drosophila melanogaster* und seines Endosymbionten *Wolbachia pipientis wMel* möglich (siehe Abbildung 3.2). Wie in Abbildung 3.6 gezeigt, ist in diesen beiden Modellen der durchschnittliche Knotengrad eines Metaboliten deutlich höher als in allen anderen Modellen. Der durchschnittliche Knotengrad der Reaktionen dieser beiden Modelle, ist wie in Abbildung 3.9 gezeigt, nicht erhöht. Das erhöhte Verhältnis basiert auf der Tatsache, dass in beiden Organismen weniger Metaboliten als üblich vorhanden sind. Bei gleichbleibendem durchschnittlichen Knotengrad der Reaktionen muss bei geringerer Anzahl an Metaboliten jeder dieser an mehr Reaktionen teilhaben als üblich, weshalb deren durchschnittlicher Knotengrad steigt. Die Genomsequenz von *Drosophila melanogaster* ist bekannt und der Organismus sehr gut untersucht (Adams et al., 2000). So bleibt die Frage offen, ob bei einem so gut untersuchten Organismus trotzdem große Lücken in den Informationen zu dessen Metabolom vorliegen oder was der Grund für dieses größere Verhältnis ist. In der Datenbank sind die Modelle von insgesamt 12 *Drosophila* Arten vorhanden, aber nur *Drosophila melanogaster* zeigt diese Auffälligkeit. Auch stellt sich die Frage, wieso dieses größere Verhältnis auch im Modell von *Wolbachia pipientis wMel* auftritt, einem Endosymbionten, welcher auch in anderen Organismen vorkommt (Wu et al., 2004).

Die Tatsache, dass Reaktionen gemittelt mehr Metaboliten verbrauchen als sie produzieren, deckt sich mit der Beobachtung, dass Metaboliten öfters als Edukt einer Reaktion auftreten als als Produkt. Eine mögliche Erklärung hierzu wurde im vorherigen Kapitel dargelegt.

## 3.4 Clusterkoeffizient von Metaboliten und Reaktionen

### 3.4.1 Clusterkoeffizient der Metaboliten

Der Clusterkoeffizient eines Metaboliten ist ein Maß dafür, wie stark dessen Nachbar-Knoten miteinander vernetzt sind. In den erzeugten PN-Modellen sind die direkten Nachbarknoten eines Metaboliten Reaktionen, und diese können in einem PN nicht mit Kanten untereinander verbunden werden. Daher erfolgt die Berechnung des Clusterkoeffizienten auf der Nachbarschaft 2. Grades eines Metaboliten $V_{\mathrm{G}}^{(2)}$, welche aus Metaboliten besteht. Abbildung 3.11 zeigt die Verteilung des Clusterkoeffizienten der Metaboliten über alle 2641 Modelle der *path2models*-Datenbank. Auf der x-Achse ist die Größe der Nachbarschaft 2. Grades eines Metaboliten $n = |V_{\mathrm{G}}^{(2)}(i)|$ aufgetragen. Die y-Achse gibt den durchschnittlichen Clusterkoeffizient $C(n)$ für einen Metaboliten mit $n$ Nachbarn 2. Grades. Die kleinste auftretende Nachbarschaft ist $n = 2$, die größte ist $n = 1188$. Der Wert für

$C(n)$ nimmt einen Bereich von $C(1109) = 0.004$ bis $C(5) = 0.28$ ein. Allgemein lässt sich beobachten, dass je größer die Nachbarschaft 2. Grades eines Metaboliten ist, desto kleiner ist dessen Clusterkoeffizient. Ein Maximum von $C(5) = 0.28$ bedeutet, dass im Durchschnitt die Hälfte aller Nachbarn durch eine Hin- oder Rückkante miteinander verbunden ist. Durch die Berechnung des Clusterkoeffizienten auf der Nachbarschaft 2. Grades eines Metaboliten steht die Größe dieser Nachbarschaft nicht mehr im direkten Verhältnis zu dessen Knotengrad. Deshalb ist der maximale Knotengrad eines Metaboliten $k = 3012$, die größte Nachbarschaft aber nur 1188. Exakte Beispiele für Metaboliten mit hohem oder niedrigen Clusterkoeffizienten lassen sich nicht geben, da die selben Metaboliten in verschiedenen Modellen auch unterschiedlich hohe Clusterkoeffizienten haben können. Der Verteilung des Clusterkoeffizienten unterliegt ein Potenzgesetz und ist somit ebenfalls skalenfrei. Nach Betrachtung der Kumulante der Verteilung, hier nicht gezeigt, kann das Potenzgesetz auf die gesamte Verteilung angewendet werden. Der $\gamma$-Wert dieser Potenzverteilung ist $\gamma = 0.92 \pm 0.0026$.

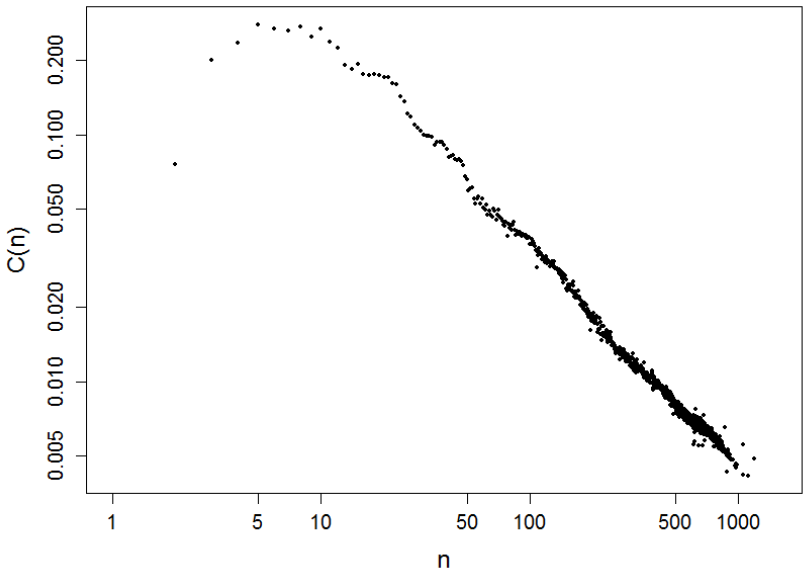


**Abbildung 3.11:** Der Plot zeigt die Verteilung des Clusterkoeffizienten der Metaboliten im Verhältnis zur Größe der Nachbarschaft 2. Grades eines Metaboliten. Auf der x-Achse ist $n$ die Größe der Nachbarschaft 2. Grades eines Metaboliten aufgetragen. Die y-Achse gibt den durchschnittlichen Clusterkoeffizienten $C(n)$ eines Metaboliten mit $n$ Nachbarn 2. Grades an. Beide Achsen sind logarithmisch skaliert. Der Wert für $C(n)$ nimmt einen Bereich von $C(1109) = 0.004$ bis $C(5) = 0.28$ ein. Die Größe der Nachbarschaft 2. Grades liegt im Bereich $1 \geq n \leq 1188$. Je größer die Nachbarschaft 2. Grades eines Metaboliten ist, desto niedriger ist dessen Clusterkoeffizient.

In Abbildung 3.12 wird die Anzahl der Metaboliten eines Modelles, x-Achse, ins Verhältnis zu deren durchschnittlichen Clusterkoeffizienten, y-Achse, gesetzt. Jeder Punkt repräsentiert hier eines der 2641 Modelle der *path2models*-Datenbank. Es lässt sich beobachten, dass bei steigender Größe eines Modells, auch der durchschnittliche Clusterkoeffizient dessen Metaboliten zunimmt. Der Wert nimmt für fast alle Modelle einen Bereich von 0.05, für

das Modell von *Hydrogenobaculum sp. 3684*, bis 0.18, für das Modell von *Methanohalophilus mahii (strain ATCC 35705 / DSM 5219 / SLP)*, ein. Ausnahmen bilden auch hier die beiden Modelle von *Drosophila melanogaster*, blauer Punkt, und *Wolbachia pipientis wMel*, orangener Punkt. Der Wert beträgt für diese beiden Modelle 0.25 beziehungsweise 0.26.

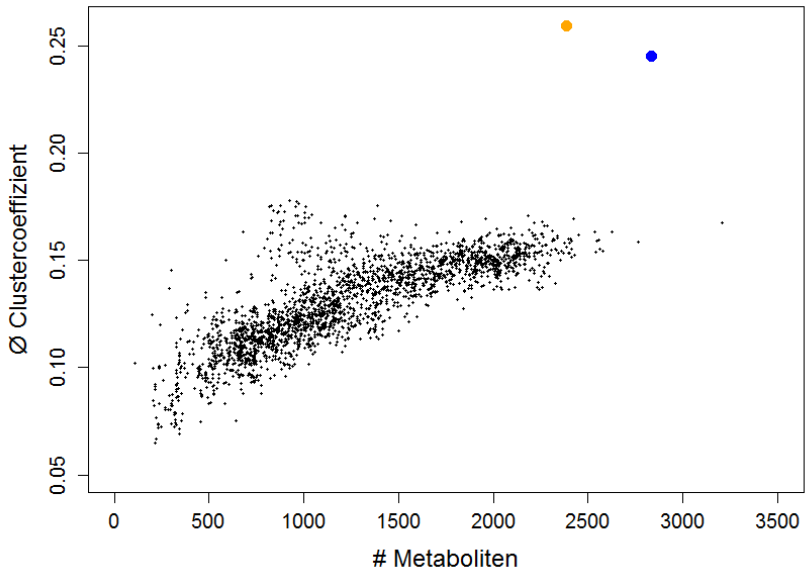


**Abbildung 3.12:** Der Plot setzt den durchschnittlichen Clusterkoeffizienten aller Metaboliten eines Modells, y-Achse, ins Verhältnis zu deren Anzahl, x-Achse. Jeder Punkt repräsentiert dabei eines der 2641 Modelle der *path2models*-Datenbank. Das Modell von *Drosophila melanogaster* wird durch einen blauen Punkt hervorgehoben, der orangene Punkt rerpäsentiert das Modell von *Wolbachia pipientis wMel*. In diesen Modellen beträgt der durchschnittliche Clusterkoeffizient aller Metaboliten 0.25 beziehungsweise 0.26. Für alle anderen Modelle nimmt der Wert einen Bereich von 0.05 bis 0.18 ein und steigt, je größer ein Modell ist.

Werden die Sekundärmetaboliten aus den Modellen entfernt, so hat dies auch einen drastischen Einfluss auf den Clusterkoeffizienten der Metaboliten. Die Verteilung des Clusterkoeffizienten nach diesem Schritt ist in Abbildung 3.13 gezeigt. Der Clusterkoeffizient der Metaboliten ist hier deutlich geringer als zuvor, der maximale Wert beträgt jetzt nur noch $C(4) = 0.06$. Zudem sinkt die maximale Größe der Nachbarschaft 2. Grades von $n = 1188$ auf $n = 28$. Erhalten bleibt jedoch der Effekt, dass der Clusterkoeffizient mit steigender Größe der Nachbarschaft 2. Grades sinkt.

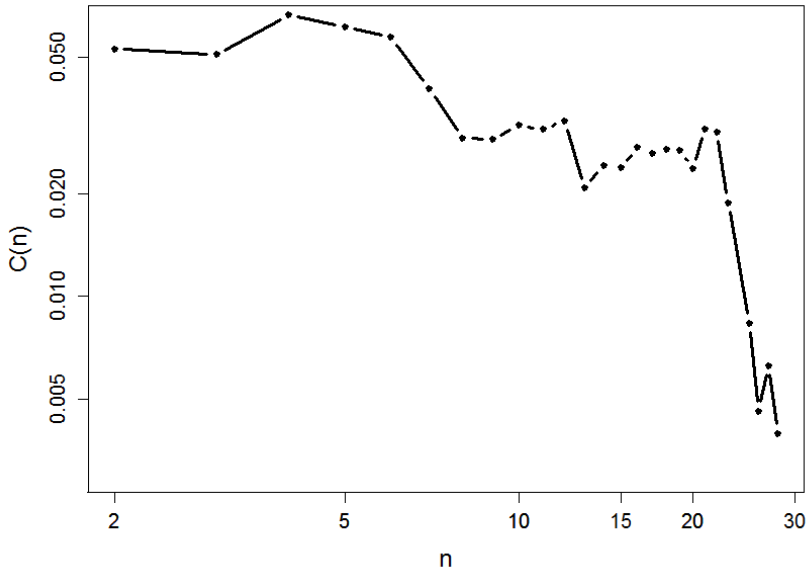


**Abbildung 3.13:** Der Plot zeigt die Verteilung des Clusterkoeffizienten der Metaboliten im Verhältnis zur Größe der Nachbarschaft 2. Grades eines Metaboliten nach Entfernen der Sekundärmetaboliten. Auf der x-Achse ist $n$ die Größe der Nachbarschaft 2. Grades eines Metaboliten aufgetragen. Die y-Achse gibt den durchschnittlichen Clusterkoeffizienten $C(n)$ eines Metaboliten mit $n$ Nachbarn 2. Grades an. Der Wert für $C(n)$ nimmt einen Bereich von $C(28) = 0.003$ bis $C(4) = 0.07$ ein. Die Größe der Nachbarschaft 2. Grades liegt im Bereich $2 \geq n \leq 28$. Je größer die Nachbarschaft 2. Grades eines Metaboliten ist, desto niedriger ist dessen Clusterkoeffizient.

### 3.4.2 Clusterkoeffizient der Reaktionen

Auch für die Reaktionen wurde der Clusterkoeffizient auf deren Nachbarschaft 2. Grades berechnet. Abbildung 3.14 zeigt die Verteilung des Clusterkoeffizienten der Reaktionen über alle 2641 Modelle der *path2models*-Datenbank. Auf der x-Achse ist die Größe der Nachbarschaft 2. Grades einer Reaktion $n = |V_{\mathrm{G}}^{(2)}|$ aufgetragen. Die y-Achse gibt den durchschnittlichen Clusterkoeffizienten $C(n)$ für eine Reaktion mit $n$ Nachbarn 2. Grades. Die Größe der Nachbarschaft 2. Grades ist im Bereich $2 \geq n \leq 2812$. Das Minimum der Verteilung liegt bei $C(183) = 0.06$, das Maximum bei $C(2102) = 0.25$ und liegt somit in einem ähnlichem Wertebereich wie bei den Metaboliten. Allgemein lässt sich hier beobachten, dass bei größerer Nachbarschaft 2. Grades auch der Clusterkoeffizient einer Reaktion steigt. Allerdings oszilliert der Clusterkoeffizient im Bereich bis $n = 600$ um den Wert $C(n) = 0.07$ bevor er steigt. Durch die Entkopplung von Knotengrad und Größe der Nachbarschaft für den Clusterkoeffizienten, ist ein interessanter Effekt zu beobachten. Beträgt der maximale Knotengrad einer Reaktion in den Modellen $k = 13$, so ist die maximale Größe der Nachbarschaft 2. Grades einer Reaktion 2812 und somit deutlich höher als für Metaboliten. Auch hier lassen sich exakte Beispiele für Reaktionen mit hohem oder niedrigem Clusterkoeffizient nicht geben.

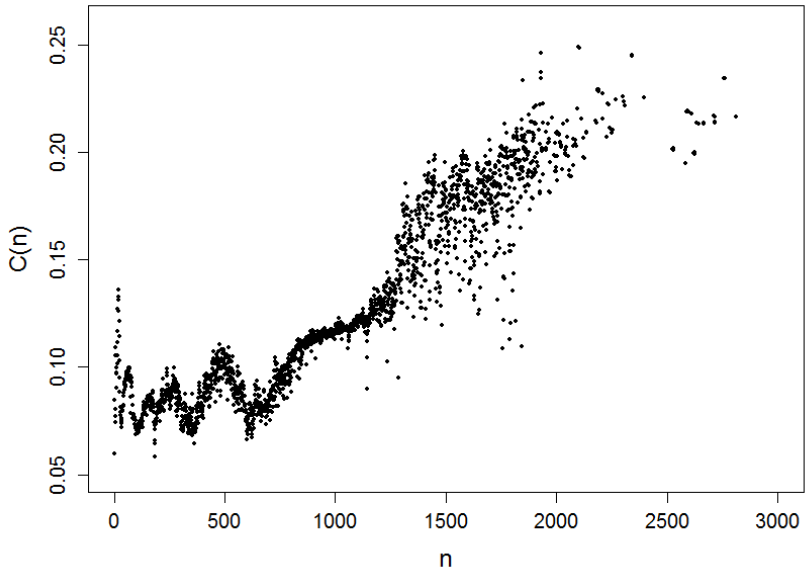


**Abbildung 3.14:** Der Plot zeigt die Verteilung des Clusterkoeffizienten der Reaktionen im Verhältnis zur Größe der Nachbarschaft 2. Grades einer Reaktion. Auf der x-Achse ist $n$ die Größe der Nachbarschaft 2. Grades einer Reaktion aufgetragen. Die y-Achse gibt den durchschnittlichen Clusterkoeffizienten $C(n)$ einer Reaktion mit $n$ Nachbarn 2. Grades an. Der Wert für $C(n)$ nimmt einen Bereich von $C(183) = 0.06$ bis $C(2102) = 0.25$ ein. Die Größe der Nachbarschaft 2. Grades liegt im Bereich $1 \geq n \leq 2812$. Der Wert für $C(n)$ oszilliert im Bereich bis $n = 600$ um den Wert $C(n) = 0.07$, für höhere Werte von $n$ steigt er an.

In Abbildung 3.15 wird die Anzahl der Reaktionen eines Modelles, x-Achse, ins Verhältnis zu deren durchschnittlichen Clusterkoeffizient, y-Achse, gesetzt. Jeder Punkt repräsentiert hier eines der 2641 Modelle der *path2models*-Datenbank. Die beiden Modelle von *Drosophila melanogaster* und *Wolbachia pipientis wMel* sind durch einen blauen, beziehungsweise orangenen Punkt hervorgehoben. Im Gegensatz zum durchschnittlichen Clusterkoeffizienten ihrer Metaboliten liegt er für ihre Reaktionen nicht deutlich höher als in den anderen Modellen. Für Modelle mit bis zu 500 Reaktionen ist der durchschnittliche Clusterkoeffizient höher als in größeren Modellen und beträgt maximal 0.15. Der maximale Wert wird für das Modell von *Nanoarchaeum equitans (strain Kin4-M)* erreicht, der niedrigste Wert 0.06 für das Modell von *Salmonella enterica subsp. enterica serovar Paratyphi B*. Allgemein lässt sich keine eindeutige Tendenz der Verteilung feststellen, da die Streuung der Werte eine Abhängigkeit des durchschnittlichen Clusterkoeffizienten von der Größe des Modells nicht zulässt.

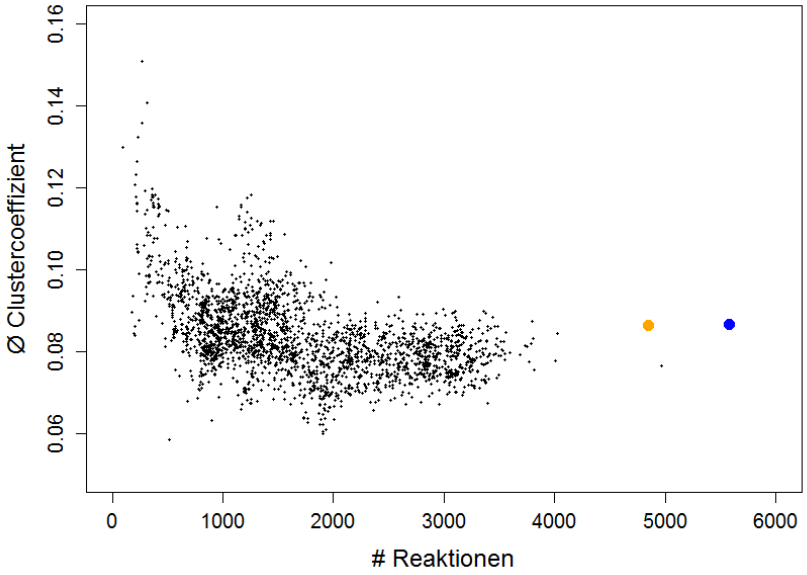


**Abbildung 3.15:** Der Plot setzt den durchschnittlichen Clustercoeffizient aller Reaktionen eines Modells, y-Achse, ins Verhältnis zu deren Anzahl, x-Achse. Jeder Punkt repräsentiert dabei eines der 2641 Modelle der *path2models*-Datenbank. Das Modell von *Drosophila melanogaster* wird durch einen blauen Punkt hervorgehoben, der orangene Punkt rerpäsentiert das Modell von *Wolbachia pipientis wMel*. Für Modelle mit bis zu 500 Reaktionen ist der durchschnittliche Clusterkoeffizient höher als in größeren Modellen und beträgt maximal 0.15. Für größere Modelle ist kein eindeutiger Trend zu beobachten. Der minimale Wert beträgt 0.06

Die Entfernung der Sekundärmetaboliten hat auch Auswirkungen auf den Clusterkoeffizienten der Reaktionen. Dieser wird in Abbildung 3.16 gezeigt. Der Clusterkoeffizient der Reaktionen ist nun deutlich geringer als zuvor, der maximale Wert beträgt jetzt nur noch $C(7) = 0.09$ und die maximale Größe der Nachbarschaft 2. Grades sinkt von $n = 2812$ auf $n = 35$. Bis auf eine kleine Spitze zwischen $7 \geq n \leq 10$ ist der Clusterkoeffizient über alle Werte für $n$ gleichbleibend gering.

Durch die Berechnung des Clusterkoeffizienten eines Knoten auf dessen Nachbarschaft 2. Grades ist die Größe dieser Nachbarschaft vom Knotengrad der Reaktion oder Metaboliten entkoppelt. Dieser Effekt wird durch die maximale Größe der Nachbarschaften 2. Grades im Vergleich zum maximalem Knotengrad deutlich. Für Metaboliten beträgt der maximale Knotengrad $k = 3012$, die größte Nachbarschaft ist aber nur $n = 1188$, also um den Faktor 2.54 kleiner. Für Reaktionen ist der maximale Knotengrad $k = 13$, die größte Nachbarschaft aber $n = 2812$, also um den Faktor 216.3 größer. Es stellt sich hier die Frage, inwiefern der Knotengrad einen Einfluss auf die Größe dieser Nachbarschaft 2. Grades hat. Um dies zu untersuchen, wurde die Korrelation des Knotengrades eines Knotens mit dessen Anzahl an Nachbarn 2. Grades berechnet. Verwendet wurde die Korrelation nach Pearson (Pearson, 1895), welche einen Korrelationskoeffizienten $r$ berechnet. Dieser Korrelationskoeffizient kann einen Wert zwischen $-1 \leq r \geq 1$ annehmen, wobei

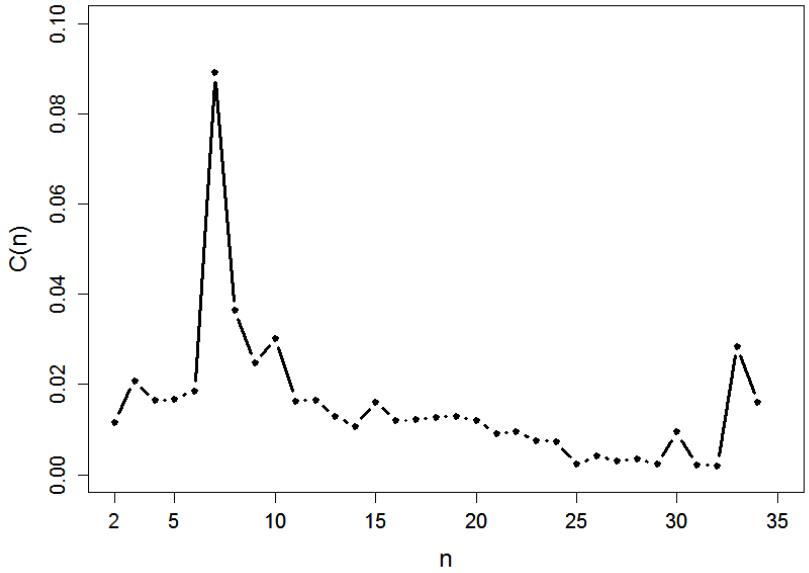


**Abbildung 3.16:** Der Plot zeigt die Verteilung des Clusterkoeffizienten der Reaktionen im Verhältnis zur Größe der Nachbarschaft 2. Grades eines Metaboliten nach Entfernen der Sekundärmetaboliten. Auf der x-Achse ist $n$ die Größe der Nachbarschaft 2. Grades eines Metabolit aufgetragen. Die y-Achse gibt den durchschnittlichen Clusterkoeffizienten $C(n)$ eines Metaboliten mit $n$ Nachbarn 2. Grades an. Der Wert für $C(n)$ nimmt einen Bereich von $C(32) = 0.002$ bis $C(7) = 0.09$ ein. Die Größe der Nachbarschaft 2. Grades liegt im Bereich $2 \geq n \leq 35$. Bis auf eine kleine Spitze zwischen $7 \geq n \leq 10$ ist der Clusterkoeffizient über alle Werte für $n$ gleichbleibend gering.

ein Wert von $r = -1$ auf eine gegenläufige Korrelation, ein Wert von $r = 0$ auf eine Unabhängigkeit der Werte und ein Wert von $r = 1$ auf eine Korrelation der Werte hinweist. Für Metaboliten ergibt sich ein Wert von $r = 0.99$ für die Korrelation von Knotengrad und Größe der Nachbarschaft 2. Grades. Betrachtet man diese für Reaktionen, so ergibt sich ein Wert von $r = 0.41$. In beiden Fällen ist bei einem höheren Knotengrad auch eine größere Nachbarschaft 2. Grades vorhanden. Der niedrigere Wert des Korrelationskoeffizienten für Reaktionen rührt wahrscheinlich von dem großen Unterschied zwischen maximalem Knotengrad und maximaler Größe der Nachbarschaft 2. Grades, die bei den Metaboliten zwar auch vorhanden ist, aber nicht ganz so groß ist, her.

Dass die Verteilung des Clusterkoeffizienten eines Netzwerkes skalenfrei ist, wenn die Verteilung des Knotengrades des Netzwerkes skalenfrei ist, wurde von Ravasz et al. (2002) gezeigt, wobei ein Wert von $\gamma \sim 1$ für das zugrundeliegende Potenzgesetz beobachtet wurde. Die in Abbildung 3.11 gezeigte Verteilung des Clusterkoeffizienten für alle Metaboliten in allen Modellen und der Wert für $\gamma = 0.92$ bestätigten diese Beobachtung. In einer weiteren Arbeit von Hao et al. (2012) wird die Skalenfreiheit dieser Verteilung durch die Anwesenheit von einigen „Super-Hubs“ erklärt, also von Knoten, mit einem sehr hohem Knotengrad, im Falle der hier untersuchten PN-Modelle die Sekundärmetaboliten. Diese Aussage kann für die untersuchten PN-Modelle beobachtet werden. Werden alle

Sekundärmetaboliten entfernt, ist, wie in Abbildung 3.13 gezeigt, die Verteilung des Clusterkoeffizienten nicht mehr skalenfrei, und die Vernetzung der Nachbarn 2. Grades eines Metaboliten bricht zusammen.

In der Arbeit von Ravasz et al. (2002) wird auch gezeigt, dass der durchschnittliche Clusterkoeffizient aller Knoten in einem Netzwerk unabhängig von dessen Größe ist. Diese Aussage ist im Falle der 2641 Modelle der *path2models*-Datenbank durch Abbildung 3.12 widerlegt. Für die untersuchten Modelle ist festzustellen, dass je mehr Metaboliten in einem Modell enthalten sind desto höher ist deren durchschnittlicher Clusterkoeffizient.

In Abbildung 3.12 stechen wieder die Modelle von *Drosophila melanogaster* und *Wolbachia pipientis wMel* heraus. Der durchschnittliche Clusterkoeffizient der Metaboliten in diesen beiden Modellen ist deutlich höher als in allen anderen Modellen. Da in diesen beiden Modellen ebenfalls der durchschnittliche Knotengrad der Metaboliten erhöht ist, haben Metaboliten in diesen Modellen eine größere Nachbarschaft 2. Grades, da deren Größe mit dem Knotengrad korreliert. Die Nachbarn eines Metaboliten haben ebenfalls einen erhöhten Knotengrad und somit viele Nachbarn, was insgesamt zu einer dichteren Vernetzung der Metaboliten in den Modellen von *Drosophila melanogaster* und *Wolbachia pipientis wMel* führt, was wiederum den höheren durchschnittlichen Clusterkoeffizienten seiner Metaboliten erklärt.

Im Gegensatz zur Verteilung des Clusterkoeffizienten der Metaboliten ist diese im Falle der Reaktionen, wie in Abbildung 3.14 zu sehen, nicht skalenfrei. Der Clusterkoeffizient steigt sogar an, je größer die Nachbarschaft 2. Grades einer Reaktion wird, also der gegenteilige Effekt wie bei Metaboliten. Sind große Nachbarschaften 2. Grades eines Metaboliten eher dünn vernetzt, so ist diese für eine Reaktion dichter vernetzt. Interessant ist hier auch die Oszillation zu Beginn der Verteilung bis zu einer Größe der Nachbarschaft 2. Grades von ungefähr 700. Die Ursache für diese Oszillation konnte bisher nicht gefunden werden und könnte in weiterführenden Arbeiten näher untersucht werden.

Der durchschnittliche Clusterkoeffizient der Metaboliten sinkt mit deren Anzahl in einem Modell. Wie in Abbildung 3.15 gezeigt, ist für Reaktionen nur bei kleineren Modellen derselbe Effekt zu beobachten. Für Modelle mit mehr als 1000 Reaktionen ist jedoch keine Abhängigkeit vom durchschnittlichen Clusterkoeffizienten und der Anzahl der Reaktionen zu beobachten. Hier trifft also die Beobachtung von Ravasz et al. (2002) im Bezug auf die Metaboliten zu.

Auch wenn die Verteilung des Clusterkoeffizienten für Reaktionen nicht skalenfrei ist, so lässt das Entfernen der Sekundärmetaboliten die Vernetzung der Nachbarschaft 2. Grades einer Reaktion, wie in Abbildung 3.16 gezeigt, stark abnehmen.

# Kapitel 4

# Zusammenfassung

Die letzten Jahrzehnte brachten einen enormen Zuwachs des Wissens und Verständnisses über die molekularen Prozesse des Lebens. Möglich wurde dieser Zuwachs durch die Entwicklung diverser Methoden, mit denen beispielsweise gezielt die Konzentration einzelner Stoffe gemessen werden kann oder gar alle anwesenden Metaboliten eines biologischen Systems erfasst werden können. Die großflächige Anwendung dieser Methoden führte zur Ansammlung vieler unterschiedlicher -*om*-Daten, wie zum Beispiel Metabolom-, Proteom- oder Transkriptoms-Datensätzen. Die Systembiologie greift auf solche Daten zurück, um mathematische Modelle biologischer Systeme zu erstellen. Eine Methode für diese Modelle ist die Modellierung des biologischen Systems als Graph. Die Erzeugung solcher Graphen und deren Untersuchung im Bezug auf graphentheoretische Fragestellungen im Kontext der Systembiologie wird im Bereich der Netzwerkbiologie zusammengeführt. Die Analyse der topologischen Eigenschaften eines Graphen ermöglicht es, grundlegende Aussagen über die globalen Eigenschaften des modellierten Systems und dessen Entstehungsprozess zu treffen und sind somit ein erster Schritt für das Verständnis eines komplexen biologischen Systems. In der Literatur sind die topologischen Eigenschaften von metabolischen Netzwerken gut untersucht, wobei hier die Netzwerke der Metaboliten und Reaktionen getrennt betrachtet werden. Motivation dieser Arbeit war es, diese beiden Netzwerke mit Hilfe des PN-Formalismus zusammenzuführen und Metaboliten und Reaktionen in einem gemeinsamen Netzwerk zu betrachten. Ziel dieser Arbeit war es zunächst zu untersuchen, ob sich die schon bekannten topologischen Eigenschaften metabolischer Systeme durch die Verwendung von PN verändern und welche neuen Eigenschaften zu beobachten sind.

Als Grundlage für die in dieser Arbeit vorgenommenen Analysen dient die *path2models* - Datenbank mit ihren 2641 Gesamtgenom-Modellen, die computergestützt aus den Informationen der KEGG- und MetaCyc-Datenbank erstellt wurden. Für die Verarbeitung dieser im SBML-Format vorliegenden Modelle wurde zunächst ein Kommandozeilenprogramm entwickelt, welches diese in PN umwandelt und deren Struktur gleichzeitig in einer Datenbank speichert. Dasselbe Programm wurde auch verwendet, um anschließend die topologischen Analysen parallelisiert auf einem Computercluster durchzuführen und deren

Ergebnisse zu speichern. Untersucht wurden im Rahmen dieser Arbeit zunächst die beiden topologischen Parameter Knotengrad und Clustercoeffizient sowie das Verhältnis der Anzahl von Reaktionen und Metaboliten in allen Modellen. Mit dem Knotengrad wird beschrieben, wie stark ein Metabolit oder eine Reaktion in das Modell eingebunden ist, also an wie vielen Reaktionen ein Metabolit beteiligt ist und wie viele Edukte und Produkte eine Reaktion hat. Der Clusterkoeffizient beschreibt, wie vernetzt die Nachbarschaft eines Metaboliten oder einer Reaktion untereinander ist. Findet man für diese Eigenschaften bestimmte Merkmale, so lässt dies auf bestimmte Eigenschaften des Modelles schließen, beispielsweise wie Robust das Modell gegen den Ausfall einzelner Elemente ist. Ein Abstecken dieser Eigenschaften für metabolische Modelle kann in Zukunft auch helfen, neu erstellte Modelle in einem ersten Schritt zu validieren.

## 4.1 Das Verhältnis der Anzahl von Reaktionen und Metaboliten

Es wurde gezeigt, dass in den 2641 verwendeten Modellen das Verhältnis der Anzahl an Reaktionen und Metaboliten korreliert. Im Durchschnitt hat ein Modell 1.57 mal mehr Reaktionen als Metaboliten. Bei steigender Anzahl an verschiedenen biochemischen Reaktionen eines Organismus steigt also auch die Diversität der von diesem Organismus umgesetzten und produzierten chemischen Verbindungen. Es werden also nicht nur neue Reaktionen zwischen den vorhandenen Metaboliten hinzugefügt.

## 4.2 Metaboliten mit sehr hohem Knotengrad

Für die Verteilung des Knotengrades und des Clusterkoeffizienten wurde gezeigt, dass diese im Bezug auf Metaboliten skalenfrei ist. Beide Effekte wurden bereits in der Literatur für monopartite Graphen gezeigt. Verantwortlich dafür sind in beiden Fällen einige wenige Metaboliten mit sehr hohem Knotengrad, wie zum Beispiel $H^+$, $H_2O$ oder $NADP^+$. Die Existenz dieser Metaboliten ist im Falle des Knotengrades die Voraussetzung für eine skalenfreie Verteilung. Werden diese Metaboliten aus den Modellen entfernt, so trifft diese Eigenschaft für die Verteilung des Clusterkoeffizienten nicht mehr zu. Diese Metaboliten sorgen zudem für die Vernetzung der Nachbarschaft von Reaktionen. Das Vorhandensein dieser wenigen Metaboliten führt zudem dazu, dass das Maximum der Verteilung der Knotengrade aller Reaktionen bei $k = 4$ und $k = 5$ liegt, da meistens zwei Metaboliten mit sehr hohem Knotengrad an einer Reaktion teilnehmen, jeweils als Edukt und als Produkt, und somit den Knotengrad von Reaktionen 1. oder 2. Ordnung erhöhen. Bei der Modellierung biologischer Systeme werden solche Metaboliten meistens nicht berücksichtigt, da sie die Komplexität des Modells erhöhen und die Durchführung von Analysemethoden erschweren können. Die Ergebnisse dieser Arbeit zeigen jedoch, dass genau diese wenigen

Metaboliten mit sehr hohem Knotengrad für die wichtigen Eigenschaften biologischer Systeme verantwortlich sind, denn eine skalenfreie Verteilung der Knotengrade oder des Clusterkoeffizienten sind zum Beispiel die Grundlage für die Robustheit dieser gegen Ausfälle einzelner Elemente (Albert et al., 2000).

## 4.3 Einfluss der Größe eines Modells auf dessen Eigenschaften

Im Gegensatz zum Verhältnis der Anzahl an Reaktionen und Metaboliten hat die Größe eines Modells in Bezug auf die beiden untersuchten topologischen Eigenschaften einen Einfluss. Steigt die Anzahl an Metaboliten in einem biologischen System, so steigt auch die Anzahl an Reaktionen, an denen diese im Durchschnitt beteiligt sind. Die durchschnittliche Anzahl an Produkten und Edukten einer Reaktion steigt jedoch nicht so stark bei größeren biologischen Systemen. Da es immer weniger Metaboliten als Reaktionen gibt, müssen bei steigender Anzahl an Reaktionen die Metaboliten an immer mehr Reaktionen beteiligt sein, um deren gleichbleibende Komplexität zu bedienen. Die Vernetzung der Nachbarschaft eines Metaboliten steigt an, je mehr Metaboliten in einem biologischem System vorhanden sind. In der Literatur (Ravasz et al., 2002) wurde jedoch beschrieben, dass der durchschnittliche Clusterkoeffizient aller Metaboliten eines biologischen Systems unabhängig von dessen Größe ist. Für die Nachbarschaft einer Reaktion ist ein umgekehrter Effekt zu erkennen, jedoch viel schwächer. Dies bestätigt die für den Knotengrad getroffene Aussage. Der schnellere Anstieg der Reaktionen beim Wachsen der Größe eines biologischen Systems, ohne dass die Komplexität der Reaktionen abnimmt, wird durch die stärkere Einbindung der einzelnen Metaboliten ermöglicht.

# Kapitel 5

# Anhang

## 5.1 Ergänzende Abbildungen

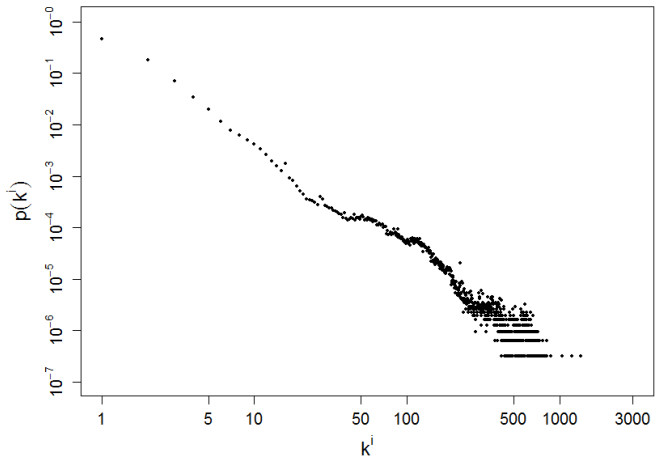


**Abbildung 5.1:** Der log-log-Plot zeigt die Verteilung der eingehenden Knotengrade der Metaboliten in allen Modellen. Ein Punkt gibt die Wahrscheinlichkeit $p(k^i)$ an, mit der ein Metabolit den eingehednen Kontengrad $k^i$ hat. Das gewichtete arithmetische Mittel der Verteilung beträgt 3.09.

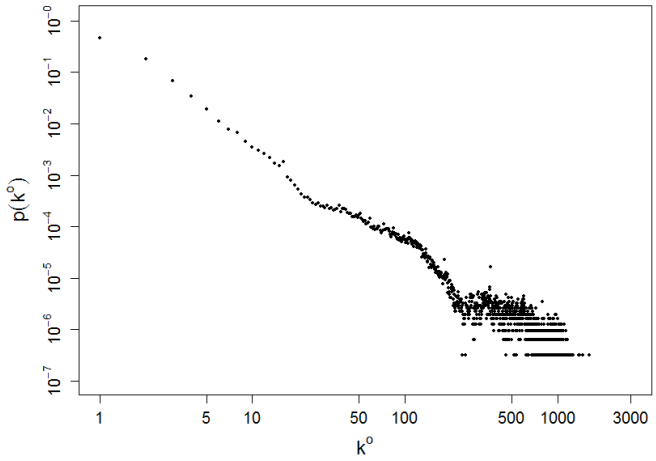


**Abbildung 5.2:** Der log-log-Plot zeigt die Verteilung der ausgehenden Knotengrade der Metaboliten in allen Modellen. Ein Punkt gibt die Wahrscheinlichkeit $p(k^o)$ an, mit der ein Metabolit den Kontengrad $k^o$ hat. Das gewichtete arithmetische Mittel der Verteilung beträgt 3.32.

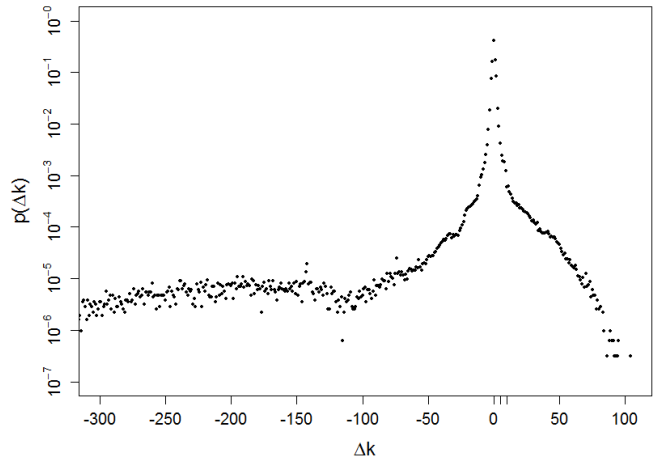


**Abbildung 5.3:** Der Plot zeigt die Wahrscheinlichkeitsverteilung von $\Delta k = k^i - k^o$ aller Metaboliten. Die logarithmierte y-Achse gibt die Wahrscheinlichkeit $p(\Delta k)$ an, mit der ein Metabolit ein bestimmtes $\Delta k$ aufweist. Der Wertebereich der x-Achse ist auf den Bereich von $-300$ bis 104 beschränkt. Im Fall $\Delta k < 0$ gilt, dass ein Metabolit von mehr Reaktionen verbraucht wird als erzeugt. Das gewichtete arithmetische Mittel der Verteilung beträgt $-0,23$.

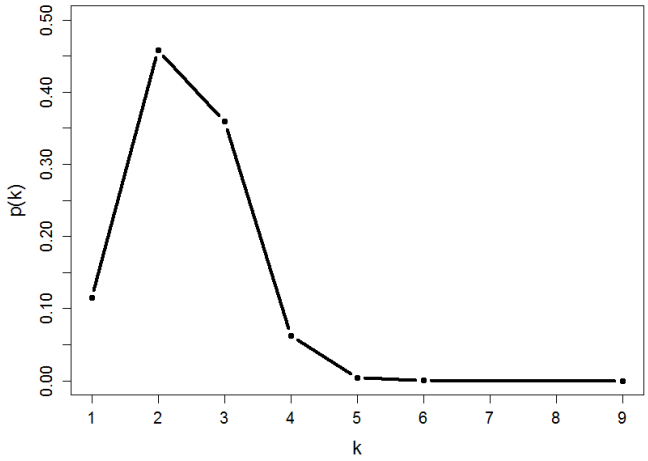


**Abbildung 5.4:** Der Plot zeigt die Verteilung der eingehenden Knotengrade der Reaktionen in allen Modellen. Ein Punkt gibt die Wahrscheinlichkeit $p(k)$ an, mit der eine Reaktion den eingehenden Kontengrad $k^i$ hat. Das gewichtete arithmetische Mittel der Verteilung beträgt 2.38.

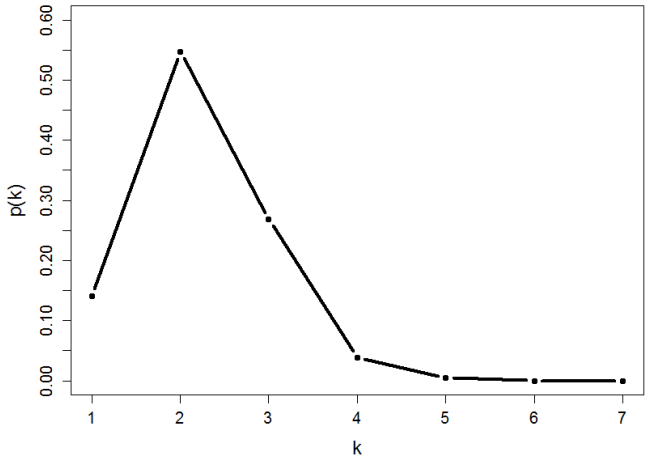


**Abbildung 5.5:** Der Plot zeigt die Verteilung der ausgehenden Knotengrade der Reaktionen in allen Modellen. Ein Punkt gibt die Wahrscheinlichkeit $p(k)$ an, mit der eine Reaktion den ausgehenden Kontengrad $k^o$ hat. Das gewichtete arithmetische Mittel der Verteilung beträgt 2.22.

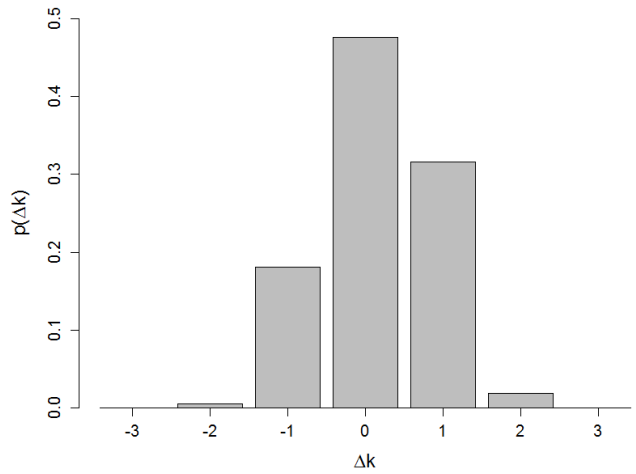


**Abbildung 5.6:** Der Plot zeigt die Wahrscheinlichkeitsverteilung von $\Delta k = k^i - k^o$ aller Reaktionen. Die y-Achse gibt die Wahrscheinlichkeit, $p(\Delta k)$, an, mit der eine Reaktion ein bestimmtes $\Delta k$ beseitzt. Der Wertebereich für $\Delta k$ ist auf Werte zwischen $-3$ und 3 beschränkt. Für den Fall $\Delta k < 0$, produziert eine Reaktion mehr Metabolite, als das sie Metaboliten verbraucht und umgekehrt. Das gewichtete arithmetische Mittel der Verteilung beträgt $0,17$.

# Literaturverzeichnis


Ackermann, J., Einloft, J., Nöthen, J., und Koch, I. (2012). Reduction techniques for network validation in systems biology. *Journal of Theoretical Biology*, 315:71–80.

Adams, M. D., Celniker, S. E., Holt, R. A., Evans, C. A., Gocayne, J. D., Amanatides, P. G., Scherer, S. E., Li, P. W., Hoskins, R. A., Galle, R. F., et al. (2000). The genome sequence of Drosophila melanogaster. *Science*, 287(5461):2185–2195.

Aebersold, R. und Mann, M. (2003). Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207.

Ahn, W. S. und Antoniewicz, M. R. (2011). Metabolic flux analysis of CHO cells at growth and non-growth phases using isotopic tracers and mass spectrometry. *Metabolic Engineering*, 13(5):598–609.

Akutsu, T., Miyano, S., und Kuhara, S. (2000). Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16(8):727–734.

Albert, I., Thakar, J., Li, S., Zhang, R., und Albert, R. (2008). Boolean network simulations for life scientists. *Source Code for Biology and Medicine*, 3(1):1–8.

Albert, R. und Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97.

Albert, R., Jeong, H., und Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *Letters to Nature*, 406(6794):378–382.

Albert, R. und Othmer, H. G. (2003). The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster. *Journal of Theoretical Biology*, 223(1):1–18.

Arita, M. (2004). The metabolic world of Escherichia coli is not small. *Proceedings of the National Academy of Sciences of the United States of America*, 101(6):1543–1547.

Artistic-License (2016). *Artistic License 2.0*. `https://opensource.org/licenses/Artistic-2.0` [Letzter Zugriff: 10. Februar 2016].

Auliac, C., Frouin, V., Gidrol, X., und d'Alché Buc, F. (2008). Evolutionary approaches for the reverse-engineering of gene regulatory networks: A study on a biologically realistic dataset. *BMC Bioinformatics*, 9(91):1–14.

Balazki, P., Lindauer, K., Einloft, J., Ackermann, J., und Koch, I. (2015). MONALISA for stochastic simulations of Petri net models of biochemical systems. *BMC Bioinformatics*, 16(215):1–11.

Barabás, A.-L. und Bonabeau, E. (2003). Scale-Free networks. *Scientific American*, 288:60–69.

Barabasi, A.-L. und Oltvai, Z. N. (2004). Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113.

Beckett, D. und McBride, B. (2004). RDF/XML Syntax Specification (Revised). *W3C Recommendation*, 1:1–56.

Binder, J. X., Pletscher-Frankild, S., Tsafou, K., Stolte, C., O'Donoghue, S. I., Schneider, R., und Jensen, L. J. (2014). COMPARTMENTS: unification and visualization of protein subcellular localization evidence. *Database*, 2014:1–9.

Bornstein, B. J., Keating, S. M., Jouraku, A., und Hucka, M. (2008). LibSBML: an API Library for SBML. *Bioinformatics*, 24(6):880–881.

Brandes, U., Eiglsperger, M., Lerner, J., und Pich, C. (2013). Graph Markup Language (GraphML). In Tamassia, R., editor, *Handbook of Graph Drawing and Visualization*. CRC Press.

Brause, F., Kemper, P., und Kritzinger, P. (1995). Abstract Petri nets notation. *Petri Net Newsletter*, 49:9–27.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., und Cowan, J. (1998). Extensible Markup Language (XML). *W3C Recommendation*, 1:1–50.

Büchel, F., Rodriguez, N., Swainston, N., Wrzodek, C., Czauderna, T., Keller, R., Mittag, F., Schubert, M., Glont, M., Golebiewski, M., van Iersel, M., Keating, S., Rall, M., Wybrow, M., Hermjakob, H., Hucka, M., Kell, D., Muller, W., Mendes, P., Zell, A., Chaouiya, C., Saez-Rodriguez, J., Schreiber, F., Laibe, C., Drager, A., und Le Novere, N. (2013). Path2Models: large-scale generation of computational models from biochemical pathway maps. *BMC Systems Biology*, 7(116):1–19.

Cao, Y. (2010). Stochastic Simulation for Biochemical Systems. In Heath, L. S. und Ramakrishnan, N., editors, *Problem Solving Handbook in Computational Biology and Bioinformatics*, pages 209–232. Springer Science.

Cao, Y., Gillespie, D. T., und Petzold, L. R. (2005). Avoiding negative populations in explicit Poisson tau-leaping. *The Journal of Chemical Physics*, 123(5):1–23.

Cao, Y., Gillespie, D. T., und Petzold, L. R. (2006). Efficient step size selection for the tau-leaping simulation method. *The Journal of Chemical Physics*, 124(4):1–11.

Caspi, R., Foerster, H., Fulcher, C. A., Kaipa, P., Krummenacker, M., Latendresse, M., Paley, S., Rhee, S. Y., Shearer, A. G., Tissier, C., Walk, T. C., Zhang, P., und Karp, P. D. (2008). The MetaCyc Database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research*, 36(suppl 1):D623–D631.

Caspi, R., Foerster, H., Fulcher, C. A., Kaipa, P., Krummenacker, M., Latendresse, M., Paley, S., Rhee, S. Y., Shearer, A. G., Tissier, C., Walk, T. C., Zhang, P., und Karp, P. D. (2014). The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research*, 42(D1):D459–471.

Cayley, A. (1874). Chemical graphs. *Philosophical Magazine*, 47:444–446.

Chaouiya, C. (2007). Petri net modelling of biological networks. *Briefings in Bioinformatics*, 8(4):210–219.

Chaouiya, C., Remy, E., und Thieffry, D. (2008). Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6(2):165–177.

Chassagnole, C., Noisommit-Rizzi, N., Schmid, J. W., Mauch, K., und Reuss, M. (2002). Dynamic modeling of the central carbon metabolism of Escherichia coli. *Biotechnology and Bioengineering*, 79(1):53–73.

Chen, M., Hariharaputran, S., Hofestädt, R., Kormeier, B., und Spangardt, S. (2011). Petri net models for the semi-automatic construction of large scale biological networks. *Natural Computing*, 10(3):1077–1097.

Chen, T., He, H. L., und Church, G. M. (1999). Modeling gene expression with differential equations. *Pacific Symposium on Biocomputing*, 4:29–40.

Chiola, G., Franceschinis, G., Gaeta, R., und Ribaudo, M. (1995). GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic Petri nets. *Performance Evaluation*, 24(1–2):47–68.

Cho, D.-Y., Kim, Y.-A., und Przytycka, T. M. (2012). Network Biology Approach to Complex Diseases. *PLoS Computational Biology*, 8(12):1–11.

Ciardo, G., Muppala, J., und Trivedi, K. (1989). SPNP: stochastic Petri net package. *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, 1:142–151.

Coffey, N. (2016). *The Numerical Recipes random number generator in Java*. `http://www.javamex.com/tutorials/random_numbers/numerical_recipes.shtml` [Letzter Zugriff: 10. Februar 2016].

Colom, J. M. und Silva, M. (1991). Convex geometry and semiflows in P/T nets.A comparative study of algorithms for computation of minimal p-semiflows. In Rozenberg, G., editor, *Advances in Petri Nets 1990*, pages 79–112. Springer Berlin Heidelberg.

Consortium, U. (2008). The Universal Protein Resource (UniProt). *Nucleic Acids Research*, 36(suppl 1):D190–D195.

Courtot, M., Juty, N., Knüpfer, C., Waltemath, D., Zhukova, A., Dräger, A., Dumontier, M., Finney, A., Golebiewski, M., Hastings, J., Hoops, S., Keating, S., Kell, D. B., Kerrien, S., Lawson, J., Lister, A., Lu, J., Machne, R., Mendes, P., Pocock, M., Rodriguez, N., Villeger, A., Wilkinson, D. J., Wimalaratne, S., Laibe, C., Hucka, M., und Le Novère, N. (2011). Controlled vocabularies and semantics in systems biology. *Molecular Systems Biology*, 7(543):1–12.

Croft, D., Mundo, A. F., Haw, R., Milacic, M., Weiser, J., Wu, G., Caudy, M., Garapati, P., Gillespie, M., Kamdar, M. R., Jassal, B., Jupe, S., Matthews, L., May, B., Palatnik, S., Rothfels, K., Shamovsky, V., Song, H., Williams, M., Birney, E., Hermjakob, H., Stein, L., und D'Eustachio, P. (2014). The Reactome pathway knowledgebase. *Nucleic Acids Research*, 42(D1):D472–D477.

Cusick, M. E., Klitgord, N., Vidal, M., und Hill, D. E. (2005). Interactome: gateway into systems biology. *Human Molecular Genetics*, 14(suppl 2):R171–R181.

DAT (2016). *Das DAT–Dateiformat*. `http://pinguin.biologie.uni-jena.de/bioinformatik/networks/metatool/metatool5.0/ecoli_networks.html` [Letzter Zugriff: 10. Februar 2016].

De Las Rivas, J. und Fontanillo, C. (2010). Protein-Protein Interactions Essentials: Key Concepts to Building and Analyzing Interactome Networks. *PLoS Computational Biology*, 6(6):1–8.

Demir, E., Babur, Ö., Rodchenkov, I., Aksoy, B. A., Fukuda, K. I., Gross, B., Sümer, O. S., Bader, G. D., und Sander, C. (2013). Using Biological Pathway Data with Paxtools. *PLoS Computational Biology*, 9(9):1–5.

Demir, E., Cary, M. P., Paley, S., Fukuda, K., Lemer, C., Vastrik, I., Wu, G., D'Eustachio, P., Schaefer, C., Luciano, J., Schacherer, F., Martinez-Flores, I., Hu, Z., Jimenez-Jacinto, V., Joshi-Tope, G., Kandasamy, K., Lopez-Fuentes, A. C., Mi, H., Pichler, E., Rodchenkov, I., Splendiani, A., Tkachev, S., Zucker, J., Gopinath, G., Rajasimha, H., Ramakrishnan, R., Shah, I., Syed, M., Anwar, N., Babur, O., Blinov, M., Brauner, E., Corwin, D., Donaldson, S., Gibbons, F., Goldberg, R., Hornbeck, P., Luna, A., Murray-Rust, P., Neumann, E., Ruebenacker, O., Samwald, M., van Iersel, M., Wimalaratne, S., Allen, K., Braun, B., Whirl-Carrillo, M., Cheung, K.-H., Dahlquist, K., Finney, A., Gillespie, M., Glass, E., Gong, L., Haw, R., Honig, M., Hubaut, O., Kane, D., Krupa, S., Kutmon, M., Leonard, J., Marks, D., Merberg, D., Petri, V., Pico, A., Ravenscroft, D., Ren, L.,

Shah, N., Sunshine, M., Tang, R., Whaley, R., Letovksy, S., Buetow, K. H., Rzhetsky, A., Schachter, V., Sobral, B. S., Dogrusoz, U., McWeeney, S., Aladjem, M., Birney, E., Collado-Vides, J., Goto, S., Hucka, M., Le Novere, N., Maltsev, N., Pandey, A., Thomas, P., Wingender, E., Karp, P. D., Sander, C., und Bader, G. D. (2010). The BioPAX community standard for pathway data sharing. *Natute Biotechnology*, 28:935–942.

Dingle, N. J., Knottenbelt, W. J., und Suto, T. (2009). PIPE2: a tool for the performance evaluation of generalised stochastic Petri Nets. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):34–39.

Einloft, J. (2016). *Visualization and analysis of functional modules in biochemical networks*. `http://www.bioinformatik.uni-frankfurt.de/tools/monalisa/index.html` [Letzter Zugriff: 10. Februar 2016].

Einloft, J., Ackermann, J., Nöthen, J., und Koch, I. (2013). MonaLisa-visualization and analysis of functional modules in biochemical networks. *Bioinformatics*, 29(11):1469–1470.

Esparza, J. (1998). Decidability and complexity of Petri net problems–an introduction. In Reisig, W. und Rozenberg, G., editors, *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*, pages 374–428. Springer Berlin Heidelberg.

exp4j (2016). *exp4j*. `http://www.objecthunter.net/exp4j/index.html` [Letzter Zugriff: 10. Februar 2016].

Feist, A. M., Herrgård, M. J., Thiele, I., Reed, J. L., und Palsson, B. Ø. (2009). Reconstruction of biochemical networks in microorganisms. *Nature Reviews Microbiology*, 7(2):129–143.

Fenn, J. B., Mann, M., Meng, C. K., Wong, S. F., und Whitehouse, C. M. (1989). Electrospray ionization for mass spectrometry of large biomolecules. *Science*, 246(4926):64–71.

Fieber, M. (2004). Design and implementation of a generic and adaptive tool for graph manipulation. Diplomarbeit, Brandenburgischen Technischen Universität Cottbus-Senftenberg.

Fields, S. und Song, O.-k. (1989). A novel genetic system to detect protein-protein interactions. *Letters to Nature*, 340:245–246.

Finney, A. und Hucka, M. (2003). Systems biology markup language: Level 2 and beyond. *Biochemical Society Transactions*, 31(6):1472–1473.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

Fourier, J. B. J. (1826). Solution d'une question particuliere du calcul des inégalités. *Nouveau Bulletin des Sciences par la Société philomatique de Paris*, 99:100.

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805.

Funahashi, A., Morohashi, M., Kitano, H., und Tanimura, N. (2003). CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *Biosilico*, 1(5):159–162.

Galagan, J. E., Minch, K., Peterson, M., Lyubetskaya, A., Azizi, E., Sweet, L., Gomes, A., Rustad, T., Dolganov, G., Glotova, I., Abeel, T., Mahwinney, C., Kennedy, A. D., Allard, R., Brabant, W., Krueger, A., Jaini, S., Honda, B., Yu, W.-H., Hickey, M. J., Zucker, J., Garay, C., Weiner, B., Sisk, P., Stolte, C., Winkler, J. K., Van de Peer, Y., Iazzetti, P., Camacho, D., Dreyfuss, J., Liu, Y., Dorhoi, A., Mollenkopf, H.-J., Drogaris, P., Lamontagne, J., Zhou, Y., Piquenot, J., Park, S. T., Raman, S., Kaufmann, S. H. E., Mohney, R. P., Chelsky, D., Moody, D. B., Sherman, D. R., und Schoolnik, G. K. (2013). The Mycobacterium tuberculosis regulatory network and hypoxia. *Nature*, 499(7457):178–183.

Giese, H., Ackermann, J., Heide, H., Bleier, L., Dröse, S., Wittig, I., Brandt, U., und Koch, I. (2015). NOVA: a software to analyze complexome profiling data. *Bioinformatics*, 31(3):440–441.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361.

Gillespie, D. T. und Petzold, L. R. (2003). Improved leap-size selection for accelerated stochastic simulation. *The Journal of Chemical Physics*, 119(16):8229–8234.

Grafahrend-Belau, E., Schreiber, F., Heiner, M., Sackmann, A., Junker, B. H., Grunwald, S., Speer, A., Winder, K., und Koch, I. (2008). Modularization of biochemical networks based on classification of Petri net t-invariants. *BMC Bioinformatics*, 9(90):1–17.

Grafahrend-Belau, E., Schreiber, F., Koschützki, D., und Junker, B. H. (2009). Flux Balance Analysis of Barley Seeds: A Computational Approach to Study Systemic Properties of Central Metabolism. *Plant Physiology*, 149(1):585–598.

Grunwald, S., Speer, A., Ackermann, J., und Koch, I. (2008). Petri net modelling of gene regulation of the Duchenne muscular dystrophy. *Biosystems*, 92(2):189–205.

Gupta, S., Bisht, S. S., Kukreti, R., Jain, S., und Brahmachari, S. K. (2007). Boolean network analysis of a neurotransmitter signaling pathway. *Journal of theoretical biology*, 244(3):463–469.

H. Genrich, R. Küffner, K. V. (2001). Executable Petri Net Models for the Analysis of Metabolic Pathways. *International Journal on Software Tools for Technology Transfer*, 3(4):394–404.

Hao, D., Ren, C., und Li, C. (2012). Revisiting the variation of clustering coefficient of biological networks suggests new modular structure. *BMC Systems Bbiology*, 6(34):1–10.

Hartmann, A., Rohn, H., Pucknat, K., und Schreiber, F. (2012). Petri nets in VANTED: Simulation of Barley Seed Metabolism. In *Proceedings of the 3rd International Workshop on Biological Processes & Petri Nets*, pages 20–28.

Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., Muthukrishnan, V., Owen, G., Turner, S., Williams, M., und Steinbeck, C. (2013). The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Research*, 41(D1):D456–D463.

Haustermann, M. (2016). *Petri Nets Tool Database*. `https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html` [Letzter Zugriff: 10. Februar 2016].

Heiner, M., Koch, I., und Will, J. (2004). Model validation of biological pathways using Petri nets–demonstrated for apoptosis. *Biosystems*, 75(1–3):15–28.

Hildebrand, M. (2015). Flexible Serialisierung von Java–Datenstrukturen für MonaLisa in Protokoll-Buffern. Masterarbeit, Goethe University Frankfurt am Main.

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., und Kummer, U. (2006). COPASI–a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., , the rest of the SBML Forum:, Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J.-H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novère, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., und Wang, J. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531.

Hülsdunk, P. (2013). Reachability analysis in biochemical Petri nets. Bachelorarbeit, Goethe University Frankfurt am Main.

Husmeier, D. (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282.

Ideker, T. und Lauffenburger, D. (2003). Building with a scaffold: emerging strategies for high-to low-level cellular modeling. *Trends in Biotechnology*, 21(6):255–262.

Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., und Sakaki, Y. (2001). A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574.

Janes, K. A. und Lauffenburger, D. A. (2013). Models of signalling networks-what cell biologists can gain from them and give to them. *Journal of Cell Science*, 126(9):1913–1921.

Jensen, K. (2013). *Coloured Petri nets: basic concepts, analysis methods and practical use*, volume 1. Springer Berlin Heidelberg, second edition.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., und Barabási, A.-L. (2000). The large-scale organization of metabolic networks. *Letters to Nature*, 407(6804):651–654.

JFreeChart (2016). *JFreeChart*. `http://www.jfree.org/jfreechart/` [Letzter Zugriff: 10. Februar 2016].

JUNG (2016). *JUNG – Java Universal Network/Graph Framework*. `http://jung.sourceforge.net/` [Letzter Zugriff: 10. Februar 2016].

Kanehisa, M. und Goto, S. (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30.

Kanehisa, M., Goto, S., Sato, Y., Kawashima, M., Furumichi, M., und Tanabe, M. (2014). Data, information, knowledge and principle: back to metabolism in KEGG. *Nucleic Acids Research*, 42(D1):D199–D205.

Katz, L. (1953). A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43.

Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467.

KGML (2016). *KGML (KEGG Markup Language)*. `http://www.kegg.jp/kegg/xml/` [Letzter Zugriff: 10. Februar 2016].

KGML-Interactions (2016). *KGML Document*. `http://www.kegg.jp/kegg/xml/docs/` [Letzter Zugriff: 10. Februar 2016].

Kitano, H. (2002). Systems Biology: A Brief Overview. *Science*, 295(5560):1662–1664.

Klamt, S. und Gilles, E. D. (2004). Minimal cut sets in biochemical reaction networks. *Bioinformatics*, 20(2):226–234.

Klamt, S., Saez-Rodriguez, J., und Gilles, E. D. (2007). Structural and functional analysis of cellular networks with CellNetAnalyzer. *BMC systems biology*, 1(2):1–13.

Klamt, S. und Stelling, J. (2002). Combinatorial Complexity of Pathway Analysis in Metabolic Networks. *Molecular Biology Reports*, 29(1-2):233–236.

Klug, W. S., Cummings, M. R., Spencer, C. A., und Palladino, M. A. (2006). *Concepts of genetics*. Prentice Hall, 8th edition.

Koch, I. und Ackermann, J. (2013). On functional module detection in metabolic networks. *Metabolites*, 3(3):673–700.

Koch, I. und Heiner, M. (2008). Petri nets. In Junker, B. H. und Schreiber, F., editors, *Analysis of Biological Networks*, Book Series on Bioinformatics, pages 139–179. John Wiley & Sons.

Koch, I., Junker, B. H., und Heiner, M. (2005). Application of Petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. *Bioinformatics*, 21(7):1219–1226.

Koch, I., Reisig, W., und Schreiber, F. (2011). *Modeling in Systems Biology: The Petri Net Approach*. Springer, Berlin / Heidelberg.

Kolpakov, F. A. (2002). BiouUML-Framework for visual modeling and simulation biological systems. In *Proceedings of the International Conference on Bioinformatics of Genome Regulation and Structure*, pages 130–133.

Kononen, J., Bubendorf, L., Kallionimeni, A., Bärlund, M., Schraml, P., Leighton, S., Torhorst, J., Mihatsch, M. J., Sauter, G., und Kallionimeni, O.-P. (1998). Tissue microarrays for high-throughput molecular profiling of tumor specimens. *Nature Medicine*, 4(7):844–847.

Koschützki, D. und Schreiber, F. (2008). Centrality analysis methods for biological networks and their application to gene regulatory networks. *Gene Regulation and Systems Biology*, 2:193–201.

Kruger, N. J., Huddleston, J. E., Le Lay, P., Brown, N. D., und Ratcliffe, R. G. (2007). Network flux analysis: impact of 13 C-substrates on metabolism in Arabidopsis thaliana cell suspension cultures. *Phytochemistry*, 68(16–18):2176–2188.

Lautenbach, K. (1973). *Exakte Bedingungen der Lebendigkeit für eine Klasse von Petri–Netzen*. Number 82 in Berichte der GMD, (In German). Gesellschaft für Mathematik und Datenverarbeitung, Sankt Augustin.

Lee, D.-Y., Zimmer, R., Lee, S. Y., und Park, S. (2006). Colored Petri net modeling and simulation of signal transduction pathways. *Metabolic Engineering*, 8(2):112–122.

Li, C., Donizelli, M., Rodriguez, N., Dharuri, H., Endler, L., Chelliah, V., Li, L., He, E., Henry, A., Stefan, M. I., Snoep, J. L., Hucka, M., Le Novère, N., und Laibe, C. (2010).

BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4(92):1–14.

Li, F., Long, T., Lu, Y., Ouyang, Q., und Tang, C. (2004a). The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences of the United States of America*, 101(14):4781–4786.

Li, S., Armstrong, C. M., Bertin, N., Ge, H., Milstein, S., Boxem, M., Vidalain, P.-O., Han, J.-D. J., Chesneau, A., Hao, T., Goldberg, D. S., Li, N., Martinez, M., Rual, J.-F., Lamesch, P., Xu, L., Tewari, M., Wong, S. L., Zhang, L. V., Berriz, G. F., Jacotot, L., Vaglio, P., Reboul, J., Hirozane-Kishikawa, T., Li, Q., Gabel, H. W., Elewa, A., Baumgartner, B., Rose, D. J., Yu, H., Bosak, S., Sequerra, R., Fraser, A., Mango, S. E., Saxton, W. M., Strome, S., van den Heuvel, S., Piano, F., Vandenhaute, J., Sardet, C., Gerstein, M., Doucette-Stamm, L., Gunsalus, K. C., Wade Harper, J., Cusick, M. E., Roth, F. P., Hill, D. E., und Vidal, M. (2004b). A Map of the Interactome Network of the Metazoan C. elegans. *Science*, 303(5657):540–543.

Lindemann, C. (1995). DSPNexpress: a software package for the efficient solution of deterministic and stochastic Petri nets. *Performance Evaluation*, 22(1):3–21.

Lunn, D. J., Thomas, A., Best, N., und Spiegelhalter, D. (2000). WinBUGS-A Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337.

MacBeath, G. und Schreiber, S. L. (2000). Printing Proteins as Microarrays for High-Throughput Function Determination. *Science*, 289(5485):1760–1763.

Machado, D., Costa, R. S., Rocha, M., Ferreira, E. C., Tidor, B., und Rocha, I. (2011). Modeling formalisms in Systems Biology. *AMB Express*, 1(45):1–14.

Mahadevan, R., Edwards, J. S., und Doyle, F. J. (2002). Dynamic Flux Balance Analysis of Diauxic Growth in Escherichia coli. *Biophysical Journal*, 83(3):1331–1340.

Mayr, E. W. (1984). An Algorithm for the General Petri Net Reachability Problem. *SIAM Journal on Computing*, 13(3):441–460.

McGuinness, D. L. und van Harmelen, F. (2004). OWL Web Ontology Language Overview. *W3C Recommendation*, 1:1–22.

Michaelis, L. und Menten, M. L. (1913). Die Kinetik der Invertinwirkung. *Biochem. z*, 49:333–369.

Milacic, M., Haw, R., Rothfels, K., Wu, G., Croft, D., Hermjakob, H., D'Eustachio, P., und Stein, L. (2012). Annotating Cancer Variants and Anti-Cancer Therapeutics in Reactome. *Cancers*, 4(4):1180–1211.

MIRIAM-Registry (2016). *MIRIAM Registry.* `http://www.ebi.ac.uk/miriam/main/collections` [Letzter Zugriff: 10. Februar 2016].

Mirzoyan, L. (2013). Bewertung von Knoten in Petrinetzen mit Hilfe von Zentralitätsmaßen. Bachelorarbeit, Goethe University Frankfurt am Main.

ML-Dokumentation (2016). *MonaLisa / Wiki / Home.* `http://sourceforge.net/p/monalisa4pn/wiki/Home/` [Letzter Zugriff: 10. Februar 2016].

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Müssel, C., Hopfensitz, M., und Kestler, H. A. (2010). BoolNet—an R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics*, 26(10):1378–1380.

Nagasaki, M., Saito, A., Jeong, E., Li, C., Kojima, K., Ikeda, E., und Miyano, S. (2010). Cell Illustrator 4.0: A Computational Platform for Systems Biology. *In silico biology*, 10(1,2):5–26.

Nesvizhskii, A. I. (2010). A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *Journal of Proteomics*, 73(11):2092–2123.

Newman, M. (2005). Power laws, Pareto distributions and Zipf`s law. *Contemporary Physics*, 46(5):323–351.

Noll, D. (2013). Biologisch relevante, graphentheoretische Eigenschaften von Petri-Netzen und deren Implementierung. Bachelorarbeit, Goethe Universität Frankfurt.

Nöthen, J. (2014). *Mathematical modeling of Arabidopsis thaliana with focus on network decomposition and reduction.* Doktorarbeit, Johann Wolfgang Goethe-Universität Frankfurt am Main.

Novere, N. L., Finney, A., Hucka, M., Bhalla, U. S., Campagne, F., Collado-Vides, J., Crampin, E. J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J. L., Spence, H. D., und Wanner, B. L. (2005). Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology*, 23(12):1509–1515.

Novere, N. L., Hucka, M., Mi, H., Moodie, S., Schreiber, F., Sorokin, A., Demir, E., Wegner, K., Aladjem, M. I., Wimalaratne, S. M., Bergman, F. T., Gauges, R., Ghazal, P., Kawaji, H., Li, L., Matsuoka, Y., Villeger, A., Boyd, S. E., Calzone, L., Courtot, M., Dogrusoz, U., Freeman, T. C., Funahashi, A., Ghosh, S., Jouraku, A., Kim, S., Kolpakov, F., Luna, A., Sahle, S., Schmidt, E., Watterson, S., Wu, G., Goryanin, I., Kell, D. B., Sander, C., Sauro, H., Snoep, J. L., Kohn, K., und Kitano, H. (2009). The Systems Biology Graphical Notation. *Nature Biotechnology*, 27(8):735–741.

Olsen, K., Reynolds, K. T., und Hoffmann, A. A. (2001). A field cage test of the effects of the endosymbiont Wolbachia on Drosophila melanogaster. *Heredity*, 86(6):731–737.

Orth, J. D., Thiele, I., und Palsson, B. Ø. (2010). What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers.

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242.

Petri, C. A. (1962). *Kommunikation mit Automaten*. PhD thesis, Universität Bonn.

Pinney, J. W., Westhead, D. R., und McConkey, G. A. (2003). Petri Net representations in systems biology. *Biochemical Society Transactions*, 31(6):1513–1515.

PNT (2016). *2.3.4 Dateiformate*. `http://www2.informatik.hu-berlin.de/lehrstuehle/automaten/ina/v2.1/node14.html` [Letzter Zugriff: 10. Februar 2016].

Poolman, M. G., Miguet, L., Sweetlove, L. J., und Fell, D. A. (2009). A Genome-Scale Metabolic Model of Arabidopsis and Some of Its Properties. *Plant Physiology*, 151(3):1570–1581.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., und Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge University Press, third edition.

Puig, O., Caspary, F., Rigaut, G., Rutz, B., Bouveret, E., Bragado-Nilsson, E., Wilm, M., und Séraphin, B. (2001). The tandem affinity purification (tap) method: a general procedure of protein complex purification. *Methods*, 24(3):218–229.

Qualifiers (2016). *BioModels.net Qualifiers*. `http://co.mbine.org/standards/qualifiers` [Letzter Zugriff: 10. Februar 2016].

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rashevsky, N. (1954). Topology and life: In search of general mathematical principles in biology and sociology. *The Bulletin of Mathematical Biophysics*, 16(4):317–348.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., und Barabási, A.-L. (2002). Hierarchical Organization of Modularity in Metabolic Networks. *Science*, 297(5586):1551–1555.

Reddy, V. N., Mavrovouniotis, M. L., und Liebman, M. N. (1993). Petri Net Representations in Metabolic Pathways. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 1:328–336.

Rodriguez, N., Thomas, A., Watanabe, L., Vazirabad, I. Y., Kofia, V., Gómez, H. F., Mittag, F., Matthes, J., Rudolph, J., Wrzodek, F., Netz, E., Diamantikos, A., Eichner, J., Keller, R., Wrzodek, C., Fröhlich, S., Lewis, N. E., Myers, C. J., Le Novère, N., Palsson, B. O., Hucka, M., und Dräger, A. (2015). JSBML 1.0: providing a smorgasbord of options to encode systems biology models. *Bioinformatics*, 31(20):3383–3386.

Rohn, H., Junker, A., Hartmann, A., Grafahrend-Belau, E., Treutler, H., Klapperstück, M., Czauderna, T., Klukas, C., und Schreiber, F. (2012). VANTED v2: a framework for systems biology applications. *BMC Systems Biology*, 6(139):1–13.

Sachs, K., Gifford, D., Jaakkola, T., Sorger, P., und Lauffenburger, D. A. (2002). Bayesian Network Approach to Cell Signaling Pathway Modeling. *Science STKE*, PE38:1–5.

Sackmann, A., Heiner, M., und Koch, I. (2006). Application of Petri net based analysis techniques to signal transduction pathways. *BMC Bioinformatics*, 7(482):1–17.

Saez-Rodriguez, J., Simeoni, L., Lindquist, J. A., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U.-U., Weismantel, R., Gilles, E. D., Klamt, S., und Schraven, B. (2007). A Logical Model Provides Insights into T Cell Receptor Signaling. *PLoS Computational Biology*, 3(8):1580–1590.

SBML-Packages (2016). *Documents/Specifications/SBML Level 3/SBML Level 3 Packages - SBML.caltech.edu*. `http://sbml.org/Documents/Specifications/SBML_Level_3/Packages` [Letzter Zugriff: 10. Februar 2016].

SBML-Software-Guide (2016). *SBML Software Guide - SBML.caltech.edu*. `http://sbml.org/SBML_Software_Guide` [Letzter Zugriff: 10. Februar 2016].

Scharf, S. (2015). Konzeption und Implementierung von JUnit–Tests für bioinformatische Anwendungen. Bachelorarbeit, Goethe University Frankfurt am Main.

Scheer, M., Grote, A., Chang, A., Schomburg, I., Munaretto, C., Rother, M., Söhngen, C., Stelzer, M., Thiele, J., und Schomburg, D. (2010). BRENDA, the enzyme information system in 2011. *Nucleic Acids Research*, 1:1–7.

Schuster, S. und Hilgetag, C. (1994). ON ELEMENTARY FLUX MODES IN BIOCHEMICAL REACTION SYSTEMS AT STEADY STATE. *Journal of Biological Systems*, 2(2):165–182.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., und Ideker, T. (2003). Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Research*, 13(11):2498–2504.

Simulator-Dokumentation (2016). *Simulation mode of MonaLisa – Documenation*. `http://www.bioinformatik.uni-frankfurt.de/tools/monalisa/dokuimg/MonaLisa_Documentation_Simulator.pdf` [Letzter Zugriff: 10. Februar 2016].

Sprenger, J., Fink, J. L., Karunaratne, S., Hanson, K., Hamilton, N. A., und Teasdale, R. D. (2008). LOCATE: a mammalian protein subcellular localization database. *Nucleic Acids Research*, 36(suppl 1):D230–D233.

Starke, P. (2003). *INA-Integrated Net Analyzer*. `http://www2.informatik.hu-berlin.de/lehrstuehle/automaten/ina/` [Letzter Zugriff: 4. März 2016].

Starke, P. H. (1990). *Analyse von Petri-Netz-Modellen*, volume 6. Springer.

Steuer, R. und López, G. Z. (2007). Global Network Properties. In Björn H. Junker, F. S., editor, *Analysis of Biological Networks*, pages 29–63. John Wiley & Sons, Inc.

Swarup, A., Lu, J., DeWoody, K. C., und Antoniewicz, M. R. (2014). Metabolic network reconstruction, growth characterization and $^{13}$ C-metabolic flux analysis of the extremophile Thermus thermophilus HB8. *Metabolic Engineering*, 24:173–180.

Sylvester, J. J. (1878). Chemistry and algebra. *Nature*, 17(432):284–.

Tanaka, R. (2005). Scale-Rich Metabolic Networks. *Physical Review Letters*, 94(16):168101–1–168101–4.

Thakar, J., Pilione, M., Kirimanjeswara, G., Harvill, E. T., und Albert, R. (2007). Modeling Systems-Level Regulation of Host Immune Responses. *PLoS Computational Biology*, 3(6):1022–1039.

Thomas, R. (1973). Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563–585.

Thormann, A., Rudolph, K., und Koch, I. (2009). TInA - (T-Invariant Analysis): A tool box for exploring pathways in biochemical systems at steady state. In *Abstract Book of GCB 2009*, pages 157–158.

Tukey, J. W. (1977). *Exploratory data analysis*. Reading, Mass.

Turner, T. E., Schnell, S., und Burrage, K. (2004). Stochastic approaches for modelling in vivo reactions. *Computational Biology and Chemistry*, 28(3):165–178.

Tyson, J. J., Chen, K. C., und Novak, B. (2003). Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Current Opinion in Cell Biology*, 15(2):221–231.

von Kamp, A. und Schuster, S. (2006). Metatool 5.0: fast and flexible elementary modes analysis. *Bioinformatics*, 22(15):1930–1931.

Wagner, A. und Fell, D. A. (2001). The small world inside large metabolic networks. *Proceedings of the Royal Society of London B: Biological Sciences*, 268(1478):1803–1810.

Wang, R.-S., Saadatpour, A., und Albert, R. (2012). Boolean modeling in systems biology: an overview of methodology and applications. *Physical Biology*, 9:1–14.

Wang, Z., Gerstein, M., und Snyder, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63.

Webb, E. C. (1992). *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.* Number Ed. 6. Academic Press.

Weber, M. und Kindler, E. (2003). The Petri Net Markup Language. In Ehrig, H., Reisig, W., Rozenberg, G., und Weber, H., editors, *Petri Net Technology for Communication-Based Systems*, volume 2472 of *Lecture Notes in Computer Science*, pages 124–144. Springer Berlin Heidelberg.

Weckwerth, W. (2007). *Metabolomics: methods and protocols*, volume 358. Humana Press Inc.

West, D. B. (2001). *Introduction to Graph Theory*, volume 2. Prentice-Hall.

Wiechert, W. (2001). $^{13}$C Metabolic Flux Analysis. *Metabolic Engineering*, 3(3):195–206.

Wilkinson, D. J. (2011). *Stochastic Modelling for Systems Biology.* CRC press, Boca Raton, second edition.

Winterbach, W., Van Mieghem, P., Reinders, M., Wang, H., und de Ridder, D. (2013). Topology of molecular interaction networks. *BMC Systems Biology*, 7(90):1–15.

Wu, M., Sun, L. V., Vamathevan, J., Riegler, M., Deboy, R., Brownlie, J. C., McGraw, E. A., Martin, W., Esser, C., Ahmadinejad, N., Wiegand, C., Madupu, R., Beanan, M. J., Brinkac, L. M., Daugherty, S. C., Durkin, A. S., Kolonay, J. F., Nelson, W. C., Mohamoud, Y., Lee, P., Berry, K., Young, M. B., Utterback, T., Weidman, J., Nierman, W. C., Paulsen, I. T., Nelson, K. E., Tettelin, H., O'Neill, S. L., und Eisen, J. A. (2004). Phylogenomics of the Reproductive Parasite Wolbachia pipientis wMel: A Streamlined Genome Overrun by Mobile Genetic Elements. *PLoS Biology*, 2(3):0327–0341.

Wuchty, S. und Stadler, P. F. (2003). Centers of complex networks. *Journal of Theoretical Biology*, 223(1):45–53.

Zevedei-Oancea, I. und Schuster, S. (2003). Topological Analysis of Metabolic Networks Based on Petri Net Theory. *In Silico Biology*, 3(3):323–345.

Zheng, J., Zhang, D., Przytycki, P. F., Zielinski, R., Capala, J., und Przytycka, T. M. (2010). SimBoolNet-a Cytoscape plugin for dynamic simulation of signaling networks. *Bioinformatics*, 26(1):141–142.