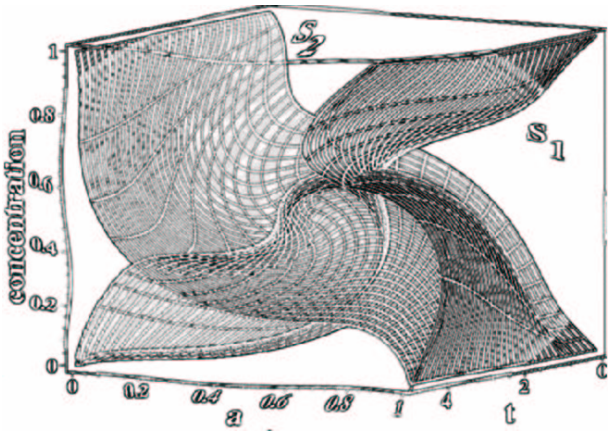
RESAC:
Eine resolutionsbasierte Künstliche Chemie und deren Anwendungen

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik
von

Jens Busch

Dortmund

2004



Tag der mündlichen Prüfung: 19.07.2004
Dekan: Prof. Dr. Bernhard Steffen
Gutachter: Prof. Dr. Wolfgang Banzhaf
Prof. Dr. Heinrich Müller

für Eun-Ok und Florian

Inhaltsverzeichnis

Einleitung

„Devices that convert information from one form into another according to a definite procedure are known as automaton“ [1]. Mit diesem Satz beginnt der im Jahre 2001 in *Nature* erschienene Artikel *Programmable and autonomous computing machine made of biomolecules* der israelischen Forschergruppe um Yaakov Benenson. Diese beschreibt die Realisierung eines endlichen Automaten[2] mit DNA-Molekülen und DNA-manipulierenden Substanzen. Die Realisierung durch autonom auf molekularer Ebene arbeitende Rechenmaschinen steckt noch in den Anfängen. Warum aber braucht man Computer, die aus Biomolekülen bestehen?

Die Grenzen der heutigen Computertechnologie werden vermutlich bald erreicht sein. Ursache dafür ist, dass die heutige Technologie ihre Weiterentwicklung hauptsächlich durch kontinuierliche Verringerung der Strukturbreite betreibt, die die Größe eines Transistors bzw. die Breite einer Leiterbahn angibt. Dafür werden zumeist lithographische Verfahren verwendet, mit denen ein ganzer Schaltplan mit mikroelektronischen Strukturen auf Siliziumchips übertragen werden kann. Bei der optischen Lithographie wird ein lichtempfindlicher Fotolack durch eine Maske belichtet. Je kleiner die Wellenlänge des verwendeten Lichts, um so höher ist die Auflösung der erzeugten Strukturen. Mit sichtbarem Licht ließen sich bald keine Fortschritte mehr erzielen. Durch UV-Licht werden heute in der Massenproduktion Strukturbreiten von 250 Nanometern (nm) erzielt, bei höherem Aufwand sogar 150 nm. Die Zukunft liegt in der Nutzung extrem ultravioletter Strahlung (EUV), deren Wellenlänge sich der der Röntgenstrahlung nähert, oder in der Anwendung neuerer Verfahren wie etwa Atom-, Ionen- oder Elektronenstrahl-Lithographie. Die erzielbaren Strukturbreiten werden für die Jahre 2008/2009 auf 32-11 nm geschätzt. Damit kommt aber die Grenze der heutigen Computertechnologie in Sicht. Die erzielte Strukturbreite wird nur noch von wenigen Atomen gebildet, so dass quantenmechanische Effekte die Funktion beeinträchtigen könnten.[3] Weitere Probleme sind die zunehmende Wärmeerzeugung pro Flächeneinheit und das Auftreten von Leckströmen.

Neben Computern aus Nano-Chips, die statt Silizium diverse Kohlenstoffverbindungen nutzen, stellen Quantencomputer und die oben erwähnten Computer aus Biomolekülen reizvolle Alternativen dar. Aus Sicht der Computerwissenschaft ist man dabei mit völlig neuen Berechnungsmus-

[1] Eine Übersetzung des Zitats aus (Benenson et al. 2001, Seite 430) lautet: Geräte, die Information nach einem genau festgelegten Verfahren von einer Form in eine andere umwandeln, werden als Automaten bezeichnet.

[2] Als endlicher Automat gilt eine Rechenmaschine, die mit einer begrenzten Zahl von Eingabesymbolen arbeitet. Durch eine Zustandsübergangsfunktion nimmt diese dabei zu jeder Zeit einen von endlich vielen internen Systemzuständen an. Der Automat bestätigt eine Eingabe, wenn er sich nach Abarbeitung dieser Eingabe in einem akzeptierenden Zustand befindet.

[3] Der Chiphersteller Intel spricht selbst von einer Grenze bei 5 nm, bei der Tunneleffekte auftreten (Zhirnov et al. 2003). Die Autoren schätzen, dass diese Grenze 2018 erreicht ist.

tern konfrontiert. So wie die von-Neumann-Rechnerarchitektur die Formulierung heutiger Algorithmen bestimmt hat, sollten die Computer der neuen Generation auch zu angepassten Berechnungsmodellen führen. Für eine effektive Nutzung wird es, wie Max Garzon und Russell Deaton (1999) feststellen, darauf ankommen, nicht nur konventionelle elektronische Implementierungen in einem neuen Medium zu emulieren, sondern einen vollständigen Paradigmenwechsel zu vollziehen. Damit gewinnen Ansätze zur Modellierung und Simulation solcher Systeme immer mehr an Bedeutung.

Im Bereich der Molekularbiologie hat man es mit einer enormen Zahl von parallel durchgeführten Rechenschritten zu tun. Diese elementaren Interaktionen zwischen Dateneinheiten, also den Molekülen, finden in einem lokal begrenzten Umfeld statt. Ein Computermodell, für das ein ähnliches Berechnungsmuster besteht, ist die so genannte *Künstliche Chemie* (KC). Sie ist daher ein Kandidat für die Modellierung von Biomolekülrechnern. Die vorliegende Arbeit entwickelt vor diesem Hintergrund eine spezielle Künstliche Chemie und zeigt deren Anwendung und Analyse.

Das Vorbild der Künstlichen Chemien ist die Chemie. Jedoch ermöglicht hier eine Abstraktion von konkreten physikalischen Randbedingungen realer chemischer Systeme und eine Abstraktion von natürlichen molekularen Prozessen eine Beschreibung dynamischer Vorgänge auf anderer Ebene. Erst die Abstraktion ermöglicht zurzeit die Simulation solcher komplexer dynamischer Systeme, denn allein die akkurate Simulation eines einzigen Moleküls bringt Computer an die Grenze ihrer Leistungsfähigkeit, wie Tim J. Hutton (2003) argumentiert. Im Modell legen Reaktionsregeln die Interaktion zwischen Molekülen fest. Die Struktur oder die Information der Moleküle kann so weitergegeben oder modifiziert werden. Eine geänderte Struktur impliziert möglicherweise eine andere Funktion. Die Rolle eines Moleküls kann sich daher mit der Zeit ändern. Das Erscheinen oder Verschwinden bestimmter Moleküle, ferner die Konzentrationsänderungen der im Reaktor befindlichen Stoffe kennzeichnen die Berechnungsschritte.

Aus der allgemeinen Form der KC leiten Forscher durch Spezifikation der Molekülmenge und der Reaktionsregelmenge Systeme ab, die Untersuchungen spezieller Fragestellungen ermöglichen. Das macht einen Vergleich wegen der unterschiedlichen Zielsetzungen schwierig. In dieser Dissertation wird eine Vereinheitlichung angestrebt. Die Struktur der Moleküle und die Reaktionsregel, von der es nur eine einzige gibt, nämlich die *Resolution*, werden festgelegt. Die daraus entstehende, auf der Resolution basierende Künstliche Chemie wird *RESAC* („resolution-based Artificial Chemistry“) genannt. Diese Festlegung führt zu einer Vereinfachung des Modells. Hat sich ein Anwender mit der Formulierung von Eingaben vertraut gemacht, ist die Umsetzung gegebener Aufgaben in das Konzept der Künstlichen Chemie erleichtert. Das gilt auch für die Analyse der Dynamik und der Ergebnisse. Das spezialisierte Modell ermöglicht insbesondere eine einfachere Anwendung im Bereich der praxisorientierten Problemlösung und vereinfacht eine Übertragung von Problemlösungsstrategien in ein verteiltes, auf massiver Parallelität beruhendes Modell.

In dieser Arbeit wird eine Molekülstruktur und eine einzige Reaktionsregel vorgeschlagen. Damit soll das allgemeine Konzept Künstlicher Chemien einschränkt werden, und zwar in einer Weise, die eine Art von intuitiver Programmierung ermöglicht, ohne das Potenzial des Paradigmas übermäßig zu schwächen. Dazu wird ein Element aus einem anderen Forschungsbereich der Künstlichen Intelligenz (KI) in das Modell eingearbeitet: die Logik. Die Prädikatenlogik 1. Ordnung definiert die Struktur der behandelten Moleküle; die von John A. Robinson (1965) entwickelte Resolutionsregel bildet die Reaktionsregel.

Teil I dieser Dissertation führt in einem Grundlagenteil in die Gebiete Prädikatenlogik und Künstliche Chemie ein, soweit diese im Rahmen der Arbeit von Bedeutung sind. Die Aussagekraft der verwendeten Logik ist dabei auch Gegenstand der Betrachtung.

Teil II präsentiert die Konstruktion und Parallelisierung des Systems. Darüber hinaus werden auch allgemeine Aspekte der KC-Forschung untersucht. Wichtiger Bestandteil der Analyse ist hier die Vorstellung eines Bereiches, in dem *RESAC* wegen ihrer Spezialisierung wesentlich geringere Modellierungsmöglichkeiten bietet als eine allgemeine Künstliche Chemie: der Bereich der *Evolutionsreaktoren*.

Teil III zeigt, wie die resolutionsbasierte Künstliche Chemie nutzbringend eingesetzt werden kann. Die erste Anwendung resultiert aus der Tatsache, dass die Resolutionsregel eine Inferenzregel[4] darstellt. Sie findet als solche Verwendung in der automatischen Deduktion. *RESAC* kann daher als automatischer Theorembeweiser eingesetzt werden. Die mechanisierte Durchführung von Beweisen ist eines der ältesten und schillerndsten Gebiete der KI. Das Theorembeweisen in einem fundierten Kalkül ist aber nicht zu verwechseln mit computergestützten Beweisen, in denen einzelne Teile durch den Computer „nachgerechnet“ werden. Einen vermeintlichen Beweis legte kürzlich der Mathematiker Thomas Hales (2002) zu einer seit 1609 unbewiesenen Vermutung von Johannes Kepler vor.[5] Mit der Hilfe von Computern überprüfte er mehr als 100.000 Ungleichungen. Doch die Gutachter der renommierten Fachzeitschrift *Annals of Mathematics* konnten die Korrektheit des Beweises nicht nachvollziehen. Allein 3 Gigabyte Datenmaterial und 250 Seiten, geschrieben im Stile eines Protokolls, veranlassten sie nach 4 Jahren zur Aufgabe. Hales versucht nun, für das Problem eine Formulierung zu finden, die eine Anwendung von automatischen Theorembeweisern ermöglicht. Bieten diese ein beweisbar korrektes Kalkül, so ist jeder einzelne Beweisschritt per definitionem korrekt. Es bleibt die Aufgabe, die Axiome der Aufgabenstellung zu validieren. Das in *RESAC* gegebene Prädikatenkalkül 1. Ordnung ist ein solch korrektes Kalkül.

Wird *RESAC* als automatischer Beweiser eingesetzt, so ist die Beweisführung ein emergenter Prozess. Das folgende Gedankenexperiment verbildlicht diesen Sachverhalt. Man denke sich einen geschlossenen Raum, in dem mehrere Millionen Menschen mit geschlossenen Augen um einen Tisch herumgehen. Eine Anzahl von Zetteln liegt auf dem Tisch. Jeder trägt eine Aussage des zu beweisenden Theorems. Unabhängig voneinander nehmen die Menschen jeweils zwei Zettel vom Tisch und versuchen, von den beiden darin enthaltenen Aussagen auf eine neue zu schließen. Gelingt dies, ersetzen sie eine der alten Aussagen durch die neue und legen beide Zettel wieder zurück auf den Tisch. Bei mehrfacher Wiederholung dieses Prozesses kann sich ein Beweis bilden, obwohl die Beteiligten über kein Problemwissen verfügen, das über die Aussagen auf beiden Zetteln hinausgeht.

Ein zweites Anwendungsgebiet von *RESAC* beschreibt die anwendungsorientierte Problemlösung. Dieser Bereich tritt in der KC-Forschung bisher etwas in den Hintergrund zu Gunsten der Modellierung.[6] Durch die Spezialisierung auf die Resolutionsregel als Reaktionsvorschrift vereinfacht *RESAC* die „Programmierung“ einer Künstlichen Chemie. Probleme können auf zweifache Arten in der resolutionsbasierten Künstlichen Chemie gelöst werden. In beiden Fällen erfolgt die

[4] Eine Inferenzregel beschreibt einen *logischen Schluss* in der Logik. Die Resolution ist eine Kombination aus Schnitt- und Spezialisierungsregel.

[5] Es geht dabei um die Vermutung, dass die dichtestmögliche Packung von Kugeln eine pyramidale Anordnung ist, bei der die einzelnen Kugeln einer Schicht in den Mulden der darunter liegenden platziert sind.

[6] (Dittrich 2001, Seite 40)

Problembeschreibung mehr deskriptiv als imperativ. Einerseits kann das Problem in prädikatenlogische Sätze umgeformt werden. Wie in der logischen Programmierung ergibt sich die Lösung hier aus dem Beweisprozess zu der Eingabe. Andererseits kann auch allein die Struktur der Logik ausgenutzt und eine Eingabe mit Blick auf die Resolutionsregel frei kreiert werden. In der vorliegenden Arbeit wird dieser Prozess anhand eines Entscheidungsproblems und eines Optimierungsproblems vorgeführt. Beide Probleme gehören der Struktur nach zur Gruppe der schwierigsten Probleme in der Informatik.

Die vorliegende Dissertation ist auch vor dem Hintergrund einer zukünftigen Realisierung auf Plattformen zu sehen, die im Gegensatz zur bestehenden Reaktorsimulation über eine hohe Informationsdichte, eine hohe Reaktionsgeschwindigkeit und über einen hohen Grad an Parallelität verfügen. Die Bildung eines enormen Rechenclusters durch Internet-Vernetzung ist schon jetzt technisch möglich. Eine Übertragung auf hochgradig parallele natürliche Systeme wie das anfangs angesprochene Rechnen mit Biomolekülen ist vorstellbar – der abschließende Ausblick geht darauf ein.

Teil I

Grundlagen

Kapitel 1

Grundlagen des Prädikatenkalküls

Die Logik ist ein leistungsfähiges Problemlösungsparadigma und überdeckt ein im letzten Vierteljahrhundert stark anwachsendes Teilgebiet der Künstlichen Intelligenz. Gleichzeitig fand in diesem Zeitraum eine intensive Entwicklung statt, Theorieentwicklung und Praxiseinsatz bedingten sich gegenseitig. Zwei Aspekte stehen bei der Wissenschaft der Logik im Vordergrund:

1. Die Wissensrepräsentation und
2. das automatische Beweisen (automatische Deduktion).

Da natürliche Sprachen zur Repräsentation von deklarativem Wissen in computerbasierten Systemen unbrauchbar sind, müssen andere Sprachen mit geeigneteren Eigenschaften benutzt werden. In der älteren angewandten Wissenschaft diente dazu meist die Sprache der Mathematik. Grundlage des Forschungsgebietes der Künstlichen Intelligenz ist heute die mathematische Logik. Diese Familie von Beschreibungssprachen für deklaratives Wissen hat im Laufe der Zeit vielfältige Facetten ausgeprägt.

Eng verbunden mit der Frage nach der Repräsentation ist die Frage, welches Potenzial eine gewählte Repräsentation in Bezug auf die Lösung von Problemen aufweist. Dies führt zum zweiten Aspekt. Das automatische Beweisen (auch automatische Deduktion genannt) beschäftigt sich mit dem Entwurf und der Implementierung von Computeralgorithmen, mit denen man in die Lage versetzt wird, mathematische Beweise zu führen. Etwas allgemeiner formuliert geht es um den so genannten Inferenzprozess, der die Ableitung von Konsequenzen aus gegebenem Wissen beschreibt. Die Beantwortung der Frage nach der Existenz eines geeigneten, d. h. vollständigen Inferenzprozesses ist nach Russell und Norvig (2003) eine der wichtigsten Resultate der Mathematik im 20. Jahrhundert.

Obwohl Kurt Gödel 1930 die Existenz einer solchen Beweisprozedur für die Prädikatenlogik 1. Ordnung zeigen konnte, dauert es noch gut 35 Jahre bis das erste praktisch anwendbare Verfahren von Robinson (1965) publiziert wurde. Der von ihm beschriebene Inferenzprozess der Resolution baut auf der gleichnamigen Inferenzregel auf und stellt ein vollständiges Beweissystem zur Verfügung. Die Inferenzregel der Resolution ist bis heute eine der wichtigsten Regeln und sehr leistungsfähig; zusammen mit ihrer Einbettung in die Prädikatenlogik steht sie in dieser Arbeit im Vordergrund.

1.1 Prädikatenlogik 1. Ordnung

> *This section sets forth the syntax of first-order logic, which is a considerably more complicated business than it was in the propositional case.*
>
> Fitting (1996, Seite 109)

Vollzieht man den Schritt von der einfachen Aussagenlogik mit ihren bloßen Aussagevariablen zu mächtigeren Beschreibungskonzepten wie der Prädikatenlogik mit elaborierten Formeln, die die Eigenschaften und Verknüpfungen von Werten über einem gewissen Interessensbereich beschreiben, so erhält man ein funktionstüchtiges Beschreibungswerkzeug von deskriptivem Wissen. Die Umsetzung der Kenntnisse einer begrenzten *Welt*, dem so genannten Weltwissen (engl. knowledge about the world), in eine formale Darstellung mit ausreichendem Beschreibungspotenzial ist der elementare und erste Schritt zur Computerrepräsentation von Wissen. Diese Umsetzung ermöglicht eine automatische, logische Deduktion von Konsequenzen aus vorliegendem Wissen.

Eine formale Sprache, die mächtig genug ist, die Formalisierung des Weltwissens durch Umsetzung einer aufgestellten *Konzeptionalisierung*[1] zu erlauben, ist die Prädikatenlogik 1. Ordnung (PL1), engl. first-order predicate logic (FOPL). Neben dem Umgang mit Objekten und ihren Beziehungen zueinander, können in dieser Logik auch Fakten über *einige* oder *alle* Objekte des Diskurses ausgedrückt werden. Im Folgenden werden nun die Konzeptionalisierung und die Grundlagen dieser am weitesten verbreiteten Logik beschrieben, und zwar getrennt nach den Aspekten Syntax und Semantik.

1.1.1 Konzeptionalisierung

Nach Genesereth und Nilsson (1987) beginnt die Formalisierung deklarativen Wissens mit einer geeigneten Konzeptionalisierung: Die *Diskurswelt* (engl. universe of discourse) beinhaltet eine nicht notwendigerweise endliche Menge von *Objekten*, über die Wissen ausgedrückt werden soll. Diese Objekte sind, ob real existent oder fiktiv, ob abstrakt oder konkret, ob elementar oder zusammengesetzt, Gegenstand der beschreiblichen Betrachtung einer zu repräsentierenden Welt. Dabei umfasst die Diskurswelt aber nicht unbedingt alle Objekte eines Weltausschnitts, sondern nur die, über die etwas auszusagen beabsichtigt ist.

Neben den Objekten enthält die Konzeptionalisierung auch die wechselseitigen Beziehungen zwischen diesen. Man unterscheidet dabei *Funktionen* und *Relationen*. n-stellige Funktionen bilden ein n-Tupel von Objekten des Diskurses auf ein anderes Objekt des Diskurses ab. Nicht alle möglichen Funktionen werden definiert, sondern nur die für eine gewünschte Beschreibung relevanten. Zusammen bilden sie die *funktionale Basismenge*.

n-stellige Relationen hingegen setzen n Objekte der Diskurswelt in eine bestimmte Relation. Dabei besteht entweder eine Beziehung zwischen den Objekten oder eben nicht. Analog zu den Funktionen wird auch hier die Menge der in einer Konzeptionalisierung betrachteten Relationen gebündelt in der *relationalen Basismenge*.

[1] Es hat sich in der deutschen Literatur eingebürgert, den Begriff der *Konzeptionalisierung* statt des Begriffes der *Konzeption* zu verwenden; es wird so die Verbindung zum engl. Fachbegriff *conceptualization* deutlich.

Eine solchermaßen definierte Konzeptionalisierung besteht dann aus dem Tripel, das durch die Objekte, die funktionale und die relationale Basismenge gebildet wird. Es stellt sich die Frage nach der Eindeutigkeit, der Güte und den Eigenschaften einer gewählten Konzeptionalisierung. Aus der hier beschriebenen Konstruktion folgt direkt, dass es unterschiedliche Konzeptionalisierungen gibt. Genesereth und Nilsson (1989) geben folgendes Beispiel aus der Welt der Physik an: Je nachdem, ob Licht als Teilchen oder als Wellenphänomen konzeptionalisiert wird, können verschiedene Aspekte des Verhaltens von Licht erklärt werden, keines reicht hier alleine aus. Die Autoren stellen weiterhin fest, dass es zurzeit keine eindeutige und ausreichende, keine erschöpfende Gütebestimmung zur Einordnung und Bewertung verschiedener Konzeptionalisierungen gibt.

Allerdings gibt es objektive Kriterien zur Unterscheidung wie die *Granularität*, die die Feinheit, sozusagen die Rastergröße der Formalisierung angibt. Ein anderer in diesem Rahmen wichtiger Aspekt ist die *Reifikation*. Dabei werden Funktionen oder Relationen zu Objekten des Diskurses gemacht (vergegenständlicht). Dies ermöglicht wiederum die Definition von Eigenschaften und Beziehungen auf Basis der gebildeten Funktions- oder Relationsobjekte.

Die Güte einer Konzeptionalisierung ist alleine bestimmt durch ihre Zweckmäßigkeit. Der erste Schritt Wissen einer Ausschnittswelt zu repräsentieren, ist, es zu konzeptionalisieren. Es gibt dafür viele Möglichkeiten, keine muss falsch sein. Jedoch werden einige zweckmäßiger sein als andere. Nach Nilsson (1998) fließt in eine Konzeptionalisierung der Erfindungsreichtum des Konzepterstellers mit ein:

> *Conceptualization usually involves an act of invention on the part of the conceptualizer.*
>
> Nilsson (1998, Seite 248)

Diese fehlende Festlegung auf eine einzige, „richtige“ Wissensrepräsentation und die unpräzise Rechtfertigung einer vorgenommenen Wahl zeigt die bekannte ontologische Unverbindlichkeit der KI. Genau diese Unverbindlichkeit wird in Kapitel 7.1 noch einmal von Interesse sein, freilich kann sie auch dort nicht aufgelöst werden.

1.1.2 Syntax

Die Syntax einer formalen Sprache legt die Regeln einer Grammatik fest, sie beschäftigt sich mit der Manipulation von Zeichenketten. In der Prädikatenlogik sind alle Symbole Zeichenketten aus Buchstaben und Ziffern, ihr Alphabet wird wie folgt definiert:

Definition 1.1.1 (Logische Symbole)
Das *Alphabet* der Prädikatenlogik 1. Ordnung besteht aus den folgenden *logischen Symbolen*:

1. Logische Operatoren: $\wedge, \vee$ und $\neg$ (auch als Überstreichung darstellbar, z. B. $\overline{F(X)}$)
2. Quantoren: $\forall, \exists$
3. Trennzeichen und Begrenzer: , ( )
4. Variablen: Die Variablenbezeichner $V_1, V_2, \ldots$ seien Elemente einer Menge $\mathcal{V}$. Im Folgenden werden sie durch mit einem Großbuchstaben beginnende Zeichenketten bezeichnet, beispielsweise sind X, Y, Var_1 Variablen aus V.

Darüber hinaus bezeichnet $\Rightarrow$ die Implikation und $\Leftrightarrow$ die Äquivalenz. Beides sind nicht-primitive logische Operatoren, es gelten für sie die definierenden Gleichungen:

$$A \Rightarrow B \;=\; (\neg A \vee B)$$

$$A \Leftrightarrow B \;=\; ((\neg A) \vee B) \wedge ((\neg B) \vee A)$$

Neben den logischen gehören die folgenden nicht-logischen Symbole zum Alphabet:

Definition 1.1.2 (Nicht-logische Symbole)
Die *nicht-logischen Symbole* des Alphabets bestehen aus:

1. Objektbezeichnern über einer zu spezifizierenden Menge $\mathcal{O}$, auch Konstantenbezeichner genannt und im Folgenden durch mit einem Kleinbuchstaben beginnende Zeichenketten bezeichnet,
2. Funktionsbezeichnern über einer zu spezifizierenden Menge $\mathcal{F}$,
3. Relationsbezeichnern über einer zu spezifizierenden Menge $\mathcal{P}$, auch Prädikatsbezeichner genannt.

Mit jeder Funktion und mit jeder Relation ist eine *Stelligkeit* verbunden, die die Zahl der Argumente der Funktion bzw. Relation angibt. So werden in Teilen der Literatur keine Objektbezeichner eingeführt, die Objekte werden dort durch Funktionen der Stelligkeit 0 dargestellt.

Die Mengen $\mathcal{P}, \mathcal{F}, \mathcal{O}$ definieren nun die formale Sprache $\mathcal{L}$ der Prädikatenlogik 1. Ordnung, notiert als Tripel $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$. Nachdem durch Def. 1.1.1 und 1.1.2 das Alphabet spezifiziert wurde, erfolgt jetzt der Aufbau der komplexeren Struktur des *Terms*. Es gibt drei Sorten von Termen:

Definition 1.1.3 (Terme)
Die Familie der *Terme* von $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ ist die kleinste Menge, für die Folgendes gilt:

1. Jede Variable ist ein Term von $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$.
2. Jeder Objektbezeichner (Element aus der Menge $\mathcal{O}$) ist ein Term von $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$.
3. Ist f eine n-stellige Funktion ($f \in \mathcal{F}$) und bezeichnen $t_1, \ldots, t_n$ Terme von $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$, so ist auch $f(t_1, \ldots, t_n)$ ein Term von $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$.

Ein Term heißt *geschlossen*, wenn er keine Variablen enthält.

Terme repräsentieren – lässig formuliert – Werte der Diskurswelt. Auf diesen aufsetzend können atomare Eigenschaften definiert und funktionale Ausdrücke zusammengesetzt werden. Man gelangt zur Definition der *wohlgeformten Sätze*, kurz Sätze genannt, engl. wellformed formulas. Es werden drei Satzformen unterschieden: *atomare, logische* und *quantifizierte* Sätze.

Definition 1.1.4 (Atomare Sätze, Logische Sätze, quantifizierte Sätze)
Ein *atomarer Satz* oder *Atom* aus $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ wird jede Anwendung $P(t_1, \ldots, t_n)$ eines n-stelligen Prädikatsymbols $P \in \mathcal{P}$ auf n Terme $t_1, \ldots, t_n$ aus $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ genannt.

Die Familie der *Sätze* $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ ist die kleinste Menge, für die Folgendes gilt:

1. Jeder atomare Satz aus $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ ist ein Satz von $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$.

2. Ist A ein Satz aus $\mathcal{L}(\mathcal{P},\mathcal{F},O)$, so ist auch $\neg A$ ein Satz aus $\mathcal{L}(\mathcal{P},\mathcal{F},O)$.
3. Sind A und B Sätze aus $\mathcal{L}(\mathcal{P},\mathcal{F},O)$, so sind auch $A\vee B, A\wedge B$ Sätze aus $\mathcal{L}(\mathcal{P},\mathcal{F},O)$.
4. Ist A ein Satz aus $\mathcal{L}(\mathcal{P},\mathcal{F},O)$ und X eine Variable aus der Menge $\mathcal{V}$, so ist auch der allquantifizierte Satz $(\forall X)A$ und der existenzquantifizierte Satz $(\exists X)A$ Satz aus $\mathcal{L}(\mathcal{P},\mathcal{F},O)$.

Die Konstrukte aus dem 2. und 3. Teil der Definition werden logische Sätze, die Konstrukte aus dem 4. Teil quantifizierte Sätze genannt.

Der letzten Definition liegt dabei implizit die Vorstellung zugrunde, dass Sätze einen bestimmten Wahrheitsgehalt aufweisen. Dieser ist jedoch *nicht* bekannt oder festsetzbar, kann aber z. B. durch logische Verknüpfungen umgeformt werden.[2]

Eine Variable kann innerhalb eines Satzes auch als Term vorkommen, ohne von einem Quantor eingeschlossen zu sein, sie heißt dann *freie Variable*. Befindet sich jedoch eine Variable innerhalb eines Geltungsbereiches eines Quantors, so wird sie *gebunden* genannt. Ein Satz ohne freie Variablen wird als *geschlossener Satz* (engl. closed formula, sentence) bezeichnet. Wenn ein Satz weder freie noch gebundene Variablen enthält, heißt er *Grundinstanz* eines Satzes.

Nachdem die syntaktischen Konstrukte der Sprache vorgestellt wurden, wird im nächsten Kapitel die Semantik formal definiert.

1.1.3 Semantik

Bei der Definition der Semantik geht es im Kern um die Festlegung, *worüber* gesprochen wird. Was definiert die *Domäne* (Domäne ist ein anderer, in diesem Zusammenhang eher gebräuchlicher Terminus für Diskurswissen bzw. Weltwissen), worüber quantifizieren die Quantoren? Wie werden die Objekt-, Funktions- und Relationsbezeichner bezüglich dieser Domäne *interpretiert*? Die Interpretation ordnet den Elementen der Sprache Elemente der Konzeptionalisierung zu. Diese beiden Dinge, die Domäne und die Interpretation werden zusammengefasst in der Definition des *Modells*, auch Struktur genannt.

Definition 1.1.5 (Modell)
Ein *Modell* für die Prädikatenlogik 1. Ordnung $\mathcal{L}(\mathcal{P},\mathcal{F},O)$ ist durch das Tupel $M=\langle D,I\rangle$ gegeben, wobei

D eine nicht-leere Menge bezeichnet, die so genannte Domäne von M und

I eine Abbildung ist, die die nicht-logischen Symbole mit Elementen der Domäne assoziiert. Diese so genannte Interpretation nimmt folgende Zuordnungen vor:

Jedem Objektbezeichner $o\in O$ wird ein Element $o^I\in D$,
jedem n-stelligen Funktionsbezeichner $f\in\mathcal{F}$ eine n-stellige Funktion $f^I: D^n\to D$,
jedem n-stelligen Prädikatsbezeichner $P\in\mathcal{P}$ eine n-stellige Relation $P^I\subseteq D^n$ zugeordnet.

[2] In manchen Büchern werden die Wahrheitswerte *wahr* und *falsch* bzw. *erfüllt* und *nicht-erfüllt* explizit zu der Gruppe der atomaren Sätze gezählt.

Im nächsten Schritt muss festgelegt werden, wofür die freien Variablen eines Satzes stehen, falls es welche gibt.

Definition 1.1.6 (Variablenzuordnung)
Eine *Variablenzuordnung* in einem Modell $\langle D, I\rangle$ ist eine Abbildung A von der Menge $\mathcal{V}$ der Variablen in die Menge D. Man spricht vom *Bild* einer Variablen $V \in \mathcal{V}$ unter der Variablenzuordnung A, bezeichnet durch V^A.

Erst Modell und Variablenzuordnung zusammen ermöglichen die Auswertung beliebiger Terme:

Definition 1.1.7 (Abbildung Term → Domäne, Termzuordnung)
Sei $M = \langle D, I\rangle$ ein Modell für $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ und A eine Variablenzuweisung in diesem Modell. Jedem Term t aus $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ kann nun ein Wert $t^{I,A}$ aus D zugewiesen werden, und zwar

1. einem Objektbezeichner o der Wert $o^{I,A} = o^I$,
2. einem Variablenbezeichner V der Wert $V^{I,A} = V^A$ und
3. einem Funktionsbezeichner f der Wert $[f(t_1, \ldots, t_n)]^{I,A} = f^I(t_1^{I,A}, \ldots, t_n^{I,A})$.

Schließlich erlaubt die Kombination von Modell und Variablenzuordnung, den Sätzen der Sprache Wahrheitswerte zuzuweisen und auf *Erfüllbarkeit* zu testen.

Definition 1.1.8 (Wahrheitswertzuordnung)
Sei $M = \langle D, I\rangle$ ein Modell für $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ und A eine Variablenzuweisung in diesem Modell. Jedem Satz Φ aus $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ wird durch $\Phi^{I,A}$ einer der Wahrheitswerte `wahr` (`w`) oder `falsch` (`f`) wie folgt zugeordnet:

1. $[P(t_1, \ldots, t_n)]^{I,A} = \mathtt{w} \;\Leftrightarrow\; (t_1^{I,A}, \ldots, t_n^{I,A}) \in P^I$
   für alle Terme $t_1, \ldots, t_n$ und n-stelligen Prädikatsbezeichner $P \in \mathcal{P}$.
2. $[\neg X]^{I,A} = \neg[X^{I,A}]$ für Sätze X.
3. $[X \circ Y]^{I,A} = X^{I,A} \circ Y^{I,A}$ für alle binären logischen Operatoren $\circ$ und Sätze X und Y.

Definition 1.1.9 (Erfüllbarkeit und verwandte Begrifflichkeiten)
Ein Satz Φ aus $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$ ist *wahr* im Modell $M = \langle D, I\rangle$ für $\mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{O})$, wenn für alle Variablenzuweisungen A gilt: $\Phi^{I,A} = \mathtt{w}$. Man schreibt dafür $M \models \Phi$.

Ein Satz Φ ist *allgemeingültig* (engl. valid), wenn Φ wahr ist in allen Modellen der Sprache. Wenn aber kein Modell existiert, in dem Φ wahr ist, so wird Φ *unerfüllbar* oder auch *inkonsistent* genannt.

Eine Menge S von Sätzen ist *erfüllbar in* $M = \langle D, I\rangle$, wenn eine Variablenzuordnung A existiert, so dass $\Phi^{I,A} = \mathtt{w}$ für alle $\Phi \in S$. S heißt *erfüllbar*, wenn es ein Modell gibt, in dem S erfüllbar ist.

Für den Umstand $M \models \Phi$ – also die Tatsache, dass ein Satz Φ wahr ist in einem Modell M – ist auch die Formulierung „M ist ein Modell *für* Φ“ gebräuchlich. Zu zeigen, welche Sätze semantisch die Folge anderer Sätze sind, ist Motivation für die Einführung des Begriffes der *logischen Implikation*, auch als logische Folge bezeichnet, engl. logical entailment.

Definition 1.1.10 (Logische Implikation)
Ein Satz Φ wird genau dann durch eine Menge S von Sätzen *logisch impliziert*, wenn Φ wahr ist in jedem Modell, in dem auch jeder Satz aus S wahr ist.

Man schreibt dafür $S \models \Phi$. Durch die in dieser Definition gewählte Formulierung wird gewissermaßen der Frage ausgewichen, wie ein fremder Betrachter bei der Überlegung, welche Konklusionen aus einer vorgelegten Satzmenge ableitbar, also wahr sind, entscheiden kann, welches die intendierte Interpretation (allgemeiner: das intendierte Modell) sein soll.[3] Der Ausweg besteht darin, nur Konklusionen abzuleiten, die wahr sind in *allen* Modellen, die die Prämissen erfüllen. Genau das geschieht bei der logischen Implikation. Sie führt zu einer weiteren grundlegenden Definition:

Definition 1.1.11 (Theorie)
Eine *Theorie* ist eine Satzmenge, die unter der logischen Implikation abgeschlossen ist.

1.2 Beweistheorie und automatische Deduktion

Es stellt sich die Frage, wie getestet werden kann, ob ein vorgelegter Satz zu einer durch eine gegebene Datenbasis induzierten Theorie gehört oder nicht. Das ist nach Def. 1.1.11 genau die Frage nach der logischen Implikation des Satzes aus der Datenbasis, auch Frage nach der *Beweisbarkeit* genannt. Betrachtet man daher Def. 1.1.10, stellt man fest, dass die darin enthaltene Unendlichkeit – die Zahl der Modelle zu einer Satzmenge ist unendlich – einen Entscheidungsalgorithmus unmöglich macht. Bei der automatischen Deduktion verlegt man daher den Schwerpunkt der Betrachtung von der Semantik auf syntaktische Aspekte und konzentriert sich auf die syntaktische *Ableitung* von Sätzen. Nach einer allgemeinen Einführung in diese Thematik wird im darauf folgenden Unterkapitel ein Beweisverfahren konkret vorgestellt.

1.2.1 Inferenzprozess

Bei dem Begriff der *Inferenz* steht die Ableitbarkeit von Sätzen aus einer Menge anderer Sätze im Mittelpunkt der Betrachtung. Gesucht sind Algorithmen, die Antworten auf beliebige in der Prädikatenlogik 1. Ordnung gestellte Fragen liefern können. Den Prozess, in dem aus Prämissen Konklusionen abgeleitet werden, nennt man *Inferenzprozess* oder auch *Beweisprozedur*. Der Inferenzprozess ist ein ein- oder mehrstufiger Prozess, in dem jeder Ableitungsschritt durch eine *Inferenzregel* – auch Schlussregel genannt – begründet wird. Ein solches System formaler Schlussregeln wird *Beweiskalkül*[4] genannt. Das Kalkül erweitert die Logik um den Begriff vom Prozess des syntaktischen Ableitens, der den semantischen Folgerungsbegriff nachbilden soll. Der allgemeine Sprachgebrauch ist bezüglich der Trennung der Begriffe Prädikatenlogik und Prädikatenkalkül, engl. predicate calculus, leider nicht einheitlich. Für die vorliegende Dissertation wird vereinbart,

[3] *wahr* gemäß Def. 1.1.9.

[4] Formal korrekt besteht ein Kalkül neben den Schlussregeln aus einer Menge von *logischen Axiomen*, die eine Minimalausstattung von allgemeingültigen Sätzen darstellen, die elementaren Tautologien. Beim Resolutionskalkül jedoch kommt ihnen keine große Bedeutung zu.

immer dann vom Prädikatenkalkül zu sprechen, wenn das entsprechende Deduktionssystem gemeint ist, also eine Menge von Inferenzregeln spezifiziert ist. Liegt dagegen die Betonung auf der Wissensrepräsentation, wird von Prädikatenlogik gesprochen.

Jede der Inferenzregeln stellt eine Konklusion einer oder mehreren Bedingungen gegenüber. Stimmen gegebene Sätze der Form nach mit den Bedingungen überein, kann in einem Schritt die zugehörige Konklusion abgeleitet werden.

Definition 1.2.1 (Ableitbarkeit)
Ist eine Menge von Inferenzregeln gegeben, so ist eine Konklusion Φ genau dann aus der Menge M der Prämissen *ableitbar*, wenn entweder Φ ein Element aus M ist, oder Φ durch Anwendung einer Inferenzregel auf das Ende einer Satzkette entsteht, in der jeder Satz aus M ableitbar ist.

Für die Ableitbarkeit eines Satzes Φ aus einer Menge M, auch *Datenbasis* genannt, engl. knowledge base, schreibt man $M \vdash \Phi$. Entsteht Φ durch einmalige Anwendung einer Inferenzregel auf M, symbolisiert man das durch $M \vdash_1 \Phi$. Weiter definiert man die Ableitung wie folgt.

Definition 1.2.2 (Ableitung)
Die Satzkette

$$M = M_1, M_2, \ldots, M_n = \Phi \text{ mit } M_i \vdash_1 M_{i+1} \text{ für } 1 \leq i < n$$

heißt die *Ableitung* von Φ aus der Datenbasis M bezüglich einer vorgegebenen Menge von Inferenzregeln.

Der Begriff der Ableitung beschreibt so einen Inferenzprozess. Die folgenden beiden Inferenzregeln sind grundlegend:

Definition 1.2.3 (Modus Ponens)
Die Inferenzregel *Modus Ponens* erlaubt die Ableitung des Satzes Ψ (Konklusion), wenn Sätze der Form $\Phi \Rightarrow \Psi$ und Φ (Prämissen) nachgewiesen sind. Man sagt, Φ wird *geschnitten*.

Definition 1.2.4 (Modus Tollens)
Die Inferenzregel *Modus Tollens* erlaubt die Ableitung des Satzes $\neg\Phi$ aus nachgewiesenen Sätzen $\Phi \Rightarrow \Psi$ und $\neg\Psi$.

Sie werden folgendermaßen visualisiert (oberhalb des Trennstriches sind die Prämissen aufgeführt, unterhalb die Konklusion):

$$\text{Modus Ponens: } \frac{\Phi \Longrightarrow \Psi \quad \Phi}{\Psi} \qquad \text{Modus Tollens: } \frac{\Phi \Longrightarrow \Psi \quad \neg\Psi}{\neg\Phi}$$

Ein wesentliches Kriterium der Inferenzregeln ist, dass diese durch rein syntaktische Kriterien gegeben sind. Sie können daher angewandt werden ohne Verständnis der Semantik der Symbole. Um die Güte von Inferenzregeln hinsichtlich ihrer Korrektheit oder ihres Potenzials beurteilen zu können, werden die beiden folgenden Gütekriterien zu ihrer Klassifikation eingesetzt. Sie beruhen auf der logischen Implikation.

Definition 1.2.5 (Konsistenz und Vollständigkeit von Inferenzregeln)
Eine Inferenzregel ist *konsistent*, wenn jeder Satz, der aus der Datenbasis abgeleitet werden kann, auch logisch durch sie impliziert wird.

Eine Inferenzregel ist *vollständig*, wenn jeder Satz, der durch die Datenbasis logisch impliziert wird, auch abgeleitet werden kann.

Mit diesen Gütekriterien kann der Begriff der Ableitung nun noch etwas restringiert werden:

Definition 1.2.6 (Beweis)
Ist die zu einer Ableitung $M \vdash \Phi$ gehörende Satzkette endlich und werden nur konsistente Inferenzregeln für ihre Konstruktion benutzt, so wird diese Ableitung *Beweis* genannt.

Wie gesehen, können neue Klauseln aus schon vorliegenden Klauseln durch Inferenzregeln hergeleitet und damit bewiesen werden. Aber kann tatsächlich die logische Implikation völlig durch eine rein syntaktische Symbolmanipulation ersetzt werden, wenn es um die Frage geht, ob ein beliebiger Satz Φ mit einer Menge M von Sätzen konsistent ist, also $M \models \Phi$ gilt? Hier hilft ein Sachverhalt, der erstmals von Gödel (1930) durch seinen berühmten Beweis über die Äquivalenz zwischen Beweis und logischer Implikation bewiesen wurde.

Satz 1.2.1 (Gödel'scher Vollständigkeitssatz (engl. completeness theorem))
Ein Satz Φ wird genau dann logisch durch eine Datenbasis M impliziert, wenn es einen Beweis dafür gibt.

Dieser wichtige Satz führt den semantischen und syntaktischen Aspekt der Prädikatenlogik zusammen. Von einer Datenbasis ausgehend (der Prämissenmenge, auch: Menge der Axiome) können Konklusionen (Theoreme) nun gleichsam abgezählt werden.

Definition 1.2.7 (Axiom, Theorem)
Inferenzregeln ohne Prämisse werden *Axiome* genannt. Sie beschreiben die grundlegenden faktischen Informationen, von denen alle anderen Schlüsse abgeleitet werden.

Jeder Satz zu einem Axiomensystem, der einen Beweis in diesem System hat, heißt *Theorem*.

Es ist zu beachten, dass der Beweis eines Theorems nicht dessen Gültigkeit nach Def. 1.1.9 zeigt, da er nur beweist, dass das Theorem in allen Modellen wahr ist, in denen auch die zugrunde liegenden Axiome wahr sind (Def. 1.1.10 und Satz 1.2.1), nicht aber in allen *möglichen* Modellen. Liegt ein Modell vor, in dem schon die Axiome nicht wahr sind, so gelten auch die abgeleiteten Theoreme nicht notwendigerweise. Theoreme entsprechen daher nicht unbedingt dem, was wir glauben oder intendieren, sondern sie folgen schlicht einer vorgegebenen Axiomenmenge (Winston 1984).

Von einer rein logischen Perspektive aus gesehen braucht eine Datenbasis daher nichts Anderes als die Theorie-aufspannenden Axiome. Insbesondere braucht sie keine Theoreme zu beinhalten, da diese die Zahl der möglichen Konklusionen nicht erhöhen und außerdem ihrerseits von den Axiomen ableitbar sind. In der Praxis aber sind Theoreme in einer Datenbasis von besonderer Bedeutung. Durch sie wird der bei der Ableitung neuer Theoreme anfallende Berechnungsaufwand zum Teil deutlich reduziert. Abbildung 1.1 verbildlicht einige der eingeführten Begriffe.

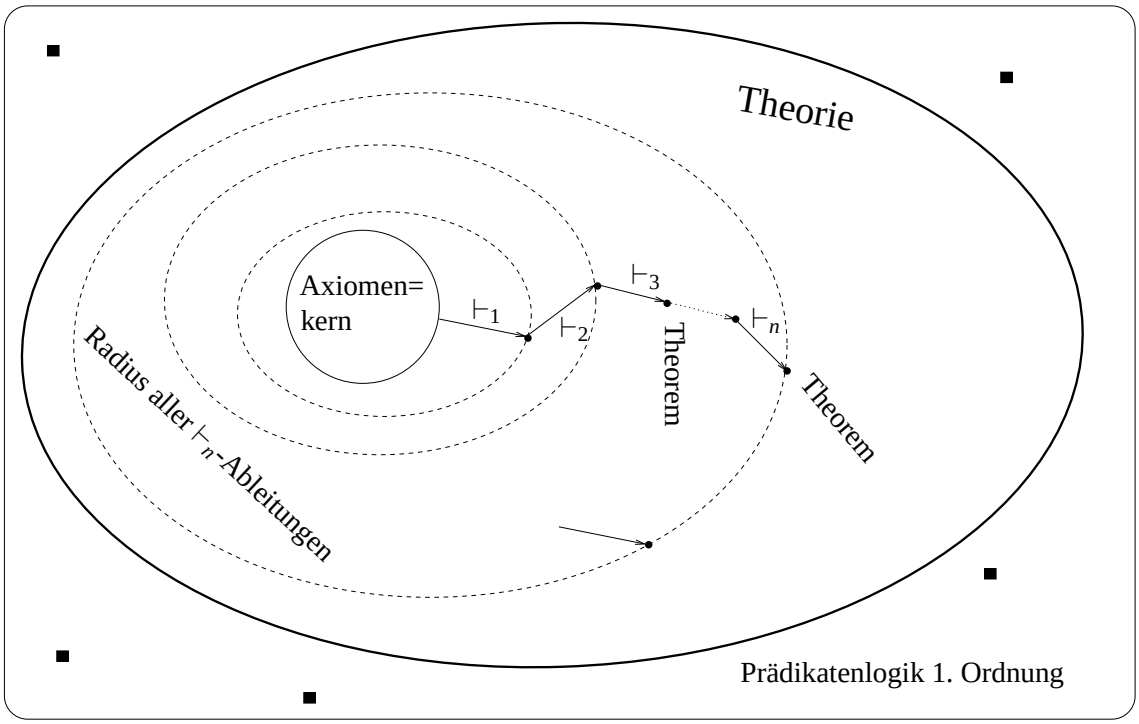


Abbildung 1.1: Verbildlichung der Begriffe Axiom, Theorem, Theorie, Ableitung. Der Axiomenkern definiert die grundlegenden Fakten. Durch syntaktische Ableitung aus den Axiomen oder vorher abgeleiteten Theoremen entstehen neue Theoreme, hier durch Punkte dargestellt. Die endliche Ableitungskette ist der Beweis. Die Menge der Theoreme bildet die Theorie. Sie ist abgeschlossen durch die logische Implikation. Quadrate symbolisieren andere Sätze der Sprache PL1.

1.2.2 Das Beweisverfahren der Resolution

John Alan Robinson entwickelte 1965 das Resolutionskalkül (Robinson 1965). Dieses war das erste durch Computer durchführbare und so in der Praxis anwendbare Beweisverfahren. Resolutionsbasierte Beweisverfahren sind auch Grundlage des Einsatzes der Logik innerhalb der *logischen Programmierung*, die durch die Gleichung *Algorithmus = Logik + Steuerung* charakterisiert werden kann (Kowalski 1979a).

Das Resolutionskalkül verwendet eine Teilsprache der Prädikatenlogik, die *Klauselform*. Seine Inferenzregel, wiederum Resolution genannt, benutzt das auf der Substitution aufbauende Verfahren der Unifikation.

Das Unterkapitel beginnt mit Einführungen zu den Themen Klauselform und Unifikation. Nach diesen Vorbetrachtungen wird die (Inferenzregel) Resolution dargestellt, gefolgt von einer Beschreibung des Inferenzprozesses. Dort wird gezeigt, wie die Resolution benutzt wird, um Theoreme zu beweisen.

Klauselform

Das Prädikatenkalkül ermöglicht durch die Anwendung der Quantoren mit vielfältigem Geltungsbereich eine beliebige Verschachtelung der Sätze. Es liegt nahe, die prädikatenlogischen Sätze in eine standardisierte Form zu bringen, die einfach zu handhaben ist in Bezug auf die gewählte Inferenzprozedur. Erreichen lässt sich dies durch die Ausnutzung tautologischer Äquivalenzen.

Voraussetzung für die Anwendung der Resolutions-Inferenzregel sind in *Klauseln* umgewandelte Sätze. Die folgende Definition führt in die Terminologie ein:

Definition 1.2.8 (Literal, Klausel, Normalformen, Klauselmenge, Klauselform)
Einfache atomare Sätze oder negierte atomare Sätze werden *Literale* genannt. Ein Literal hat *positive Polarität*, wenn der zugrunde liegende Satz nicht negiert ist, andernfalls weist es eine *negative Polarität* auf.

Die *Klausel* bezeichnet die disjunktive Verkettung ein oder mehrerer Literale.

Stehen alle Klauseln $(L_{1,1} \vee \ldots \vee L_{1,n}), \ldots, (L_{k,1} \vee \ldots \vee L_{k,m})$ einer *Klauselmenge M* in konjunktiver Verbindung, so liegt M in *konjunktiver Normalform (KNF)* vor, wobei die $L_{i,j}$ Literale bezeichnen. Die *disjunktive Normalform (DNF)* bezeichnet dagegen die disjunktive Verbindung $(L_{1,1} \wedge \ldots \wedge L_{1,n}) \vee \ldots \vee (L_{k,1} \wedge \ldots \wedge L_{k,m})$ – sie ist im vorliegenden Zusammenhang jedoch ohne Relevanz.

Die Mengendarstellung der konjunktiven Normalform heißt *Klauselform* oder *Klauselsprache*.

Um Axiome in Klauselform umzuwandeln, sind neun Schritte nötig:

1. Entfernung von Implikationen

Ersetzung von $\Rightarrow, \Leftarrow$ und $\Leftrightarrow$ durch äquivalente Sätze mit Operatoren $\neg, \vee$ und $\wedge$.

- $A \Rightarrow B \equiv \neg A \vee B$.
- $A \Leftarrow B \equiv A \vee \neg B$.
- $A \Leftrightarrow B \equiv (\neg A \vee B) \wedge (A \vee \neg B)$.

2. Verschiebung von Negationen zu den Atomen.

Benötigt werden dazu die folgenden Äquivalenzen:

- $\neg(A \wedge B) \equiv \neg A \vee \neg B$ (De Morgan).
- $\neg(A \vee B) \equiv \neg A \wedge \neg B$ (De Morgan).
- $\neg(\neg A) \equiv A$.
- $\neg\forall X[A(X)] \equiv \exists X[\neg A(X)]$.
- $\neg\exists X[A(X)] \equiv \forall X[\neg A(X)]$.

3. Variablenumbenennung.

Umbenennung von Variablen in Sätzen, in denen gleiche Variablennamen in verschiedenen Quantor-Geltungsbereichen auftreten. Beispiel:

$$(\forall X\, P(X)) \vee (\exists X\, Q(X)) \equiv (\forall X\, P(X)) \vee (\exists Y\, Q(Y))$$

4. Entfernung von Existenzquantoren.

Dieser Prozess heißt *Skolemisierung* (Skolem 1920). Ein Satz $\forall X_1 \ldots X_n \exists Y \; \mathcal{F}$, wobei in $\mathcal{F}$ kein Quantor mehr vorkommt, wird umgeformt in $\forall X_1 \ldots X_n \; \mathcal{F}^*$. $\mathcal{F}^*$ entsteht aus $\mathcal{F}$ durch Ersetzung aller freien Vorkommen von Y durch die *Skolemfunktion* $f_Y(X_1, \ldots, X_n)$. Die neu eingeführte Funktion repräsentiert gerade ein Y, das zu gegebenen $X_1, \ldots, X_n$ existieren soll. Im Allgemeinen bestimmen also die Allquantoren im Geltungsbereich des zu eliminierenden Existenzquantors die Argumente der Skolemfunktionen.

5. Verschiebung der Allquantoren nach links.

Zu diesem Zeitpunkt sind bereits alle Variablen eindeutig und allquantifiziert. Es können durch die Verschiebung keine Missverständnisse entstehen.

6. Verschiebung der Disjunktionen zu den Literalen.

Dies wird durch folgende Äquivalenz gerechtfertigt:

$$A \vee (B \wedge C) \;\equiv\; (A \vee B) \wedge (A \vee C)$$

7. Entfernung von Konjunktionen.

Die im letzten Schritt entstandenen Konjunktionen werden nun als eine Menge von Klauseln geschrieben. So wird jede Konjunktion zu einem Axiom – jeder Teil einer Konjunktion muss ja erfüllt werden.

8. Variablenumbenennung.

Umbenennung von Variablen, so dass die Variablen verschiedener Klauseln disjunkt sind. Diesen Prozess nennt man *Variablen standardisieren.*

9. Entfernung der Allquantoren.

Es gilt die Vereinbarung, dass in der Klauselsprache alle Variablen allquantifiziert angenommen werden.

Die aufgelisteten Umwandlungsschritte können einfach automatisiert werden. Die Anwendung dieser Prozedur auf eine Satzmenge S der Prädikatenlogik 1. Ordnung wird bezeichnet durch $\langle S \rangle_K$.

Die Klauselform erscheint sehr restriktiv. Das ist jedoch nicht der Fall. Zwar ist die Umwandlung in Klauselform nicht modellerhaltend (was an der Skolemisierung liegt), für jeden Satz der Prädikatenlogik gibt es aber eine in Bezug auf Erfüllbarkeit äquivalente Klauselmenge. Der Satz ist genau dann erfüllbar, wenn die entsprechende Klauselmenge erfüllbar ist. Andersherum – und dies wird beim Resolutionsprozess (siehe unten) ausgenutzt – gilt entsprechend, dass ein Satz in KNF unerfüllbar ist, wenn schon der Satz in der Originalformulierung unerfüllbar war (Loveland 1978).

Abschließend erfolgt die Definition einiger spezieller Klauseln:

Definition 1.2.9 (Horn-Klausel, unitäre Klausel und leere Klausel)

Klauseln mit höchstens einem positiven Literal heißen *Horn-Klauseln*, Klauseln, die nur aus einem Literal bestehen *unitäre* Klauseln (engl. unit-clause). Besteht dagegen die Klausel aus

gar keinem Literal, wird sie als *leere Klausel* (engl. empty clause) bezeichnet und durch $\square$ notiert. Diese ist unerfüllbar, da sie unter keiner Interpretation erfüllbar ist. Sie zeigt daher einen Widerspruch an.

Substitution und Unifikation

Voraussetzung für eine Anwendung der Resolution ist eine erfolgreiche *Unifikation*, die unterschiedliche logische Ausdrücke anzugleichen versucht. Der Begriff der *Substitution* ist dafür grundlegend.

Definition 1.2.10 (Substitution, Umbenennung)
Unter einer Substitution versteht man jede endliche Menge von Zuordnungen zwischen Variablen und Termen, wobei die Zuordnungen zwei Bedingungen erfüllen:

(1) Jeder Variable wird höchstens ein Term zugeordnet.

(2) Innerhalb einem einer Variablen zugeordneten Term tritt diese Variable nicht auf (*Occurcheck*).

Eine Substitution σ heißt *Umbenennung*, engl. renaming, genau dann, wenn durch σ Variablen nur durch Variablen ersetzt werden. Dabei darf jede Variable nur einmal zugeordnet werden (Injektivität): Für Variablen X, Y mit $X \neq Y$ gilt: $X\sigma \neq Y\sigma$. Es handelt sich also um eine Variablenpermutation.

Wird Variable X durch den Term t ersetzt, schreibt man X/t.[5] Die ersetzenden Terme kann man auch als Bindungen der zu ersetzenden Variablen auffassen. Eine *Anwendung* der Substitution σ auf einen Satz Φ erzeugt einen neuen Satz Ψ, $\Psi = \Phi\sigma$, in dem alle gebundenen Variablen durch ihre Bindungen ersetzt wurden. Ungebundene Variablen bleiben unverändert.

Die Inferenzregel Resolution setzt Substitutionen voraus, die unterschiedliche logische Ausdrücke ähnlich erscheinen lassen. Solche Substitutionen heißen *Unifikatoren*.

Definition 1.2.11 (Unifikator, allgemeinster Unifikator)
Eine Substitution θ heißt *Unifikator* einer Menge von Termen $D = \{t_1, \ldots, t_n\}$ genau dann, wenn $t_1\theta = t_2\theta = \ldots = t_n\theta$.

σ heißt *allgemeinster Unifikator* (engl. most general unifier, Abk. MGU) genau dann, wenn für jeden anderen Unifikator θ eine Substitution γ existiert, so dass gilt: $\theta = \sigma\gamma$.

Der MGU stellt auf diese Weise so etwas wie den einfachsten Unifikator dar. Eine wichtige Eigenschaft des allgemeinsten Unifikators ist seine Eindeutigkeit bis auf Variablenumbenennung, wie Robinson zeigen konnte. Eine Diskussion der Unifikationsalgorithmen, der Algorithmen also, die zu zwei gegebenen Ausdrücken einen allgemeinsten Unifikator berechnen, erfolgt an anderer Stelle (siehe Unterkapitel 1.3.1).

[5] In der Literatur findet man auch die Notation t/X mit der Sprechweise „Term t ersetzt Variable X“.

Binäre Resolutionsregel

In den Def. 1.2.3 und 1.2.4 wurden schon zwei Inferenzregeln betrachtet. In diesem Kapitel wird nun die zentrale Inferenzregel des automatischen Beweisens innerhalb des Prädikatenkalküls eingeführt. Die Inferenzregel Resolution – abgekürzt *Resolutionsregel* – ist die heutzutage wohl weit verbreiteste formale Schlussregel. Die Resolutionsregel ist eine Verallgemeinerung des Modus Ponens aus Def. 1.2.3: Auf alles, worauf mit dem Modus Ponens geschlossen werden kann, kann auch mit der Resolution geschlossen werden. Darüber hinaus ist aber die Anwendung von Substitutionen vorgesehen, um Literale der Inferenz durch Modus Ponens zugänglich zu machen. Es sind dabei jedoch nur solche Substitutionen gestattet, die nötig sind, um einen Schnitt gemäß Def. 1.2.3 ansetzen zu können.

Der Grundgedanke der Resolution ist für den Fall, dass die beteiligten Sätze keine Variablen enthalten, nicht kompliziert: Ist bekannt, dass erstens *A* wahr oder *B* wahr ist und zweitens, dass *A* falsch oder *C* wahr ist, dann kann gefolgert werden, dass entweder *B* oder *C* wahr ist. Im allgemeinen, variablenbehafteten Fall besteht die Aufgabe darin, zwei komplementäre Literale zu finden, die durch einen allgemeinsten Unifikator angeglichen werden können:

Definition 1.2.12 (Inferenzregel binäre Resolution)

Es seien zwei Klauseln Φ und Ψ gegeben. Die binäre Resolutionsregel kann genau dann angewendet werden, wenn es ein Literal ϕ in Φ und ein Literal $\neg\psi$ in Ψ gibt, so dass ϕ und ψ einen allgemeinsten Unifikator γ besitzen. Ist die Resolution durchführbar, beschreibt die Vereinigung von Φ und Ψ unter Auslassung der komplementären Literale und nach Anwendung der Substitution γ das Ergebnis der Inferenz, das *Resolvente* genannt wird.

Die binäre Resolution kann folgendermaßen visualisiert werden (oberhalb des Trennstriches sind die Prämissen aufgeführt, unterhalb die Konklusion):

Φ	mit $\phi \in \Phi$
Ψ	mit $\neg\psi \in \Psi$
$\phi\gamma = \psi\gamma$	
$((\Phi - \{\phi\}) \cup (\Psi - \{\neg\psi\}))\gamma$	

Im Folgenden wird nicht weiter betont, dass die Resolution zwischen zwei Klauseln stattfindet – daher ihr Name *binäre* Resolution – und nur noch von *Resolution* gesprochen.

Bemerkung: Wichtig ist, dass die Variablenbezeichner der an der Resolution beteiligten Klauseln disjunkt sind, die Variablen also entsprechend umbenannt werden. Dieser Prozess wird *Standardisierung der Variablen* genannt und ändert nichts am Wahrheitsgehalt der Klauseln.

Beispiel 1.2.1 (Verdeutlichung der Resolution)
(1) Wenn John etwas besitzt, so ist dies keine Katze.
(2) Pussy ist eine Katze.

Daraus entstehen die beiden Sätze in Prädikatenlogik:

(1') $\forall$`X: besitzt(john, X)` $\Rightarrow$ $\overline{\texttt{Katze(X)}}$
(2') `Katze(pussy)`.

Die Konvertierung in Klauselform liefert:

(1") $\{\overline{\texttt{besitzt(john, X)}}, \overline{\texttt{Katze(X)}}\}$
(2") $\{\texttt{Katze(pussy)}\}$.

Eine Inferenz durch Resolutionsanwendung zwischen Klauseln (1") und (2") resolviert die Literale $\overline{\texttt{Katze(X)}}$ und `Katze(pussy)` unter dem allgemeinsten Unifikator $\{\texttt{X/pussy}\}$. Abgeleitet wird die Klausel

$$\overline{\texttt{besitzt(john, X)}}\ \{\texttt{X/pussy}\} = \overline{\texttt{besitzt(john, pussy)}}.$$

Das entspricht dem Satz

$$\texttt{besitzt(john, pussy)} \Rightarrow \texttt{falsch}.$$

Mit anderen Worten: John besitzt Pussy nicht.

Im Allgemeinen können zwei Klauseln, die miteinander resolvieren, mehrere Resolventen besitzen, denn es kann mehrere Möglichkeiten geben, ϕ und ψ zu wählen. Andererseits können zwei Klauseln höchstens endlich viele Resolventen besitzen.

Ein Beweiskalkül, das als einzige Inferenzregel die Resolution nach Def. 1.2.12 enthielte, wäre jedoch nicht vollständig. Es fehlt eine Inferenzregel, die redundante Information innerhalb einer Klausel verdichtet, beispielsweise mehrere Kopien eines Literals einer Klausel entfernt. Diesen Prozess nennt man Faktorisierung, dabei wird solche redundante Information durch Unifikatoren aufgespürt.

Definition 1.2.13 (Faktorisierung)
Besitzt eine Teilmenge von Literalen einer Klausel Φ einen allgemeinsten Unifikator γ, so nennt man die Klausel, die durch Anwendung von γ auf Φ entsteht, einen *Faktor* von Φ. Die Verschmelzung von Faktoren wird *Faktorisierung*[6], engl. factoring, genannt.

Beispiel: Faktorisierung der in Klauselform vorliegenden Klausel $\{\texttt{P(f(Y)),P(X),R(X,Y)}\}$ ergibt mit dem allgemeinsten Unifikator $\{\texttt{X/f(Y)}\}$ die Klausel $\{\texttt{P(f(Y)),R(f(Y),Y)}\}$.

Bemerkung: Robinson hatte diese Verschmelzungsregel in die Resolutionsregel integriert. Dadurch wird die Definition allerdings ein wenig unübersichtlich und eine Aufspaltung in Resolution und Faktorisierung hat sich eingebürgert. Es ist zu beachten, dass aus diesem Grund berechtigterweise auch von *einer* Regel gesprochen werden kann. Die Faktorisierung wird daher in der Literatur oft in einem mit „Resolution“ überschriebenen Kapitel behandelt.

Durch Anwendung der Resolution entstehen wieder allgemeingültige Klauseln. Insbesondere erzeugt die Resolution zweier einelementiger Klauseln eine leere Ergebnisklausel, die aus keinem Literal besteht. Ein solcher Fall zeigt die Inkonsistenz der zugrunde liegenden Datenbasis an; die Resolution zwischen P und $\neg P$ ist ein Beispiel dafür.

[6] Man findet gelegentlich auch den Ausdruck *Faktorisation* in der Literatur.

Das Resolutionskalkül und seine Eigenschaften

Nach den Vorbetrachtungen der letzten Kapitel kann nun das Resolutionskalkül definiert werden:

Definition 1.2.14 (Resolutionskalkül)
Das *Resolutionskalkül R* ist gegeben durch:

(i) Sprache: Klauselsprache,

(ii) Inferenzregeln: Resolutionsregel kombiniert mit Faktorisierung,

(iii) Ableitung: Eine Ableitung (oder Herleitung) in *R*, genannt *R-Ableitung*, ist eine Ableitung gemäß Def. 1.2.2 mit der Inferenzregel aus (ii). Es bezeichnet $S \vdash_R C$ die *R*-Ableitung einer Klausel *C* aus einer Klauselmenge *S*.

Eine *R-Widerlegung* einer Klauselmenge *S* ist eine *R-Herleitung* der leeren Klausel $\square$ aus *S*.

Zusätzlich werden bei der *R*-Ableitung und der *R*-Widerlegung alle Klauseln variablenfremd angenommen, verschiedene Klauseln enthalten also keine gemeinsamen Variablensymbole. Durch Variablenumbenennung kann dies immer erreicht werden, wenn Variablen durch bisher noch nicht verwendete Variablen ersetzt werden. Da Klauseln durch äquivalente Klauseln ersetzt werden, ist dieses Verfahren korrekt.

Folgende Eigenschaften weist das Resolutionskalkül auf (die Beweise sind z. B. in Loveland (1978) zu finden), es bezeichne dabei $\mathcal{K}$ die Menge aller Klauseln:

1. Für alle $S \subset \mathcal{K}$ und für alle $C \in \mathcal{K}$ gilt: wenn $S \vdash_R C$, dann $S \models C$. Das Resolutionskalkül ist daher *korrekt*.

2. Es existieren $S \subset \mathcal{K}$ und $C \in \mathcal{K}$ mit $S \models C$ und $S \nvdash_R C$, d. h. *R* ist *unvollständig*.

3. Für alle $S \subset \mathcal{K}$ gilt: wenn $S \models \square$, dann $S \vdash_R \square$. Das Resolutionskalkül ist also *widerlegungsvollständig*.

Erscheint die 2. Eigenschaft auf den ersten Blick nachteilig, so liegt doch gerade hier der substanzielle Fortschritt des Resolutionskalküls gegenüber klassischen Kalkülen in Hinblick auf die Anwendbarkeit von Deduktionssystemen. Durch *R* sind nicht mehr alle, doch genügend viele „interessante“ Folgerungen ableitbar. Ein Kalkül, das korrekt und vollständig ist, muss nach Görtz et al. (2000) ungeeignet zur Implementierung eines automatischen Beweisers sein. Die Ursache liegt in der Ableitung aller *Tautologien* und aller *Spezialisierungen*. Tautologien sind allgemeingültige Sätze der Art $A \Rightarrow A, \square \Rightarrow A, \phi \vee \psi \vee \neg\phi$. Genauso lassen sich durch Spezialisierungen unendlich viele Sätze herleiten: beispielsweise folgt aus ϕ auch $\phi \vee \phi_1$ oder $\phi \vee [\phi_1 \vee \phi_2]$.

Das (unvollständige) Resolutionskalkül *R* hat nicht den Nachteil vollständiger Kalküle. Es ist weder möglich Sätze abzuleiten, die unabhängig von der gegebenen Axiomenmenge gelten, noch ist es möglich, zu einer gegebenen Klausel beliebig viele weitere Klauseln durch Literalerweiterung abzuleiten. Es wird dadurch ein großer Teil des Suchraums, in dem eine Suche nicht lohnte, gar

nicht erst aufgespannt.[7] Trotzdem kann man doch noch einen korrekten und vollständigen automatischen Beweiser konstruieren: Sei Φ ein Satz und S eine Menge von Sätzen der Prädikatenlogik 1. Ordnung.

$$\begin{aligned}
\text{Es gilt:} \quad & S \models \Phi && (1.1)\\
\text{gdw.} \quad & S \cup \{\neg\Phi\} \models \Box, && \text{mit Def. 1.1.9,} \quad (1.2)\\
\text{gdw.} \quad & \langle S \cup \{\neg\Phi\}\rangle_K \models \Box, && \text{mit Kap. 1.2.2 (Klauselform),} \quad (1.3)\\
\text{gdw.} \quad & \langle S \cup \{\neg\Phi\}\rangle_K \vdash_R \Box, && \text{mit Eigenschaften (1) und (3).} \quad (1.4)
\end{aligned}$$

Das nächste Unterkapitel zeigt, wie diese Beziehung in eine Beweisprozedur umgesetzt wird.

Der Resolutionsprozess zum Theorembeweis

Soll gezeigt werden, dass eine Satzmenge S den Satz Φ logisch impliziert, so kann das durch die Inkonsistenz (Unerfüllbarkeit) der Satzmenge $S \cup \neg\Phi$ gezeigt werden (*Widerlegungstheorem*, ausgenutzt in (1.2)). In diesem Fall kann die leere Klausel durch das Resolutionskalkül abgeleitet werden (1.4). Entscheidend sind die im vorigen Unterkapitel angeführten Eigenschaften (1) und (3), die die Korrektheit und Widerlegungsvollständigkeit der Resolution zeigen. Resolution beweist Theoreme daher durch Widerlegung.

Definition 1.2.15 (Resolutionswiderlegung)
Die Methode, die logische Implikation über den Nachweis der Unerfüllbarkeit durch Ableitung der leeren Klausel im Resolutionskalkül nachzuweisen, wird *Resolutionswiderlegung* genannt.

Die Resolutionswiderlegung wird folgendermaßen algorithmisch beschrieben:

Algorithmus 1.1 (Algorithmus zur Resolutionswiderlegung in R)

1. Das zu beweisende Theorem wird negiert der Liste der Axiome zugefügt.
2. Die Liste der Axiome wird in Klauselform umgewandelt (siehe Kap. 1.2.2).
3. Suche nach resolvierbaren Klauseln und Anwendung der Resolution auf diese,

   solange bis leere Klausel die erzeugte Resolvente ist oder kein resolvierbares Klauselpaar mehr existiert.
4. Wurde die leere Klausel erzeugt: Ausgabe „Theorem wahr“, andernfalls „Theorem falsch“.

Algorithmus 1.1 arbeitet wie jeder automatischer Beweiser (Görtz et al. 2000) nach dem Aufzählungsprinzip, d. h. es wird die Menge aller Sätze S^R aufgezählt, die in R aus der Axiomenmenge S (angereichert mit dem negierten Theorem Φ) hergeleitet werden können. Ist die Ausgabe „Theorem wahr“, so ist die Antwort verlässlich, das Theorem wird tatsächlich durch die ursprüngliche

[7] Hier wird der Gegensatz zu anderen Kalkülen der Prädikatenlogik, den *Gentzen-Kalkülen* (Gentzen 1935) deutlich, in denen die angesprochenen Ableitungen möglich sind. Es lässt sich eine enorme Zahl von Ableitungsmöglichkeiten nicht verhindern.

Axiomenmenge impliziert. Darüber hinaus gilt auch die Umkehrung, es liegt also eine Äquivalenzbeziehung vor.

Gilt andererseits $S \not\models \Phi$, so hält Algorithmus 1.1 niemals mit der Ausgabe „Theorem wahr“, denn er arbeitet nach den Vorbetrachtungen korrekt. Es gibt allerdings keine Garantie, dass er hält. Diese wichtige Eigenschaft ist kein spezielles Problem der Resolutionswiderlegung. Eine fundamentale Erkenntnis der Berechenbarkeitstheorie von Alonzo Church und (unabhängig im gleichen Jahr publiziert) Alan Turing besagt, dass eine Entscheidungsprozedur prinzipiell nicht erreicht werden kann.

Satz 1.2.2 (Church (1936), Turing (1936))
Die Prädikatenlogik 1. Ordnung ist unentscheidbar.

Der Grund liegt darin, dass alle vollständigen Beweisprozeduren für die Prädikatenlogik erster Ordnung auf das Halteproblem zurückgeführt werden können.[8] Die Menge der allgemeingültigen Sätze der Prädikatenlogik 1. Ordnung ist rekursiv aufzählbar, die komplementäre Menge der falsifizierbaren Sätze jedoch nicht.

Das Beste, was ein automatischer Beweiser prinzipiell leisten kann, ist die *Semientscheidbarkeit*: Wenn ein Satz wahr ist, kann das auch in endlich vielen Schritten bewiesen werden. Andernfalls jedoch gibt es keine Garantie, dass der Theorembeweiser terminiert. Wenn er aber terminiert, ist seine Aussage verlässlich. Der Algorithmus 1.1 zur Resolutionswiderlegung ist ein semientscheidbares Verfahren, da er – wie angesprochen – genau dann auf „Theorem wahr“ entscheidet, wenn der getestete Ausdruck wirklich ein Theorem ist.

1.3 Weitere Aspekte des Resolutionsprozesses

1.3.1 Unifikationsalgorithmen

Unifikationsalgorithmen bestimmen den allgemeinsten Unifikator (siehe Def. 1.2.11) zu zwei Atomen oder Termen, im Folgenden zusammengefasst unter der Bezeichnung *Unifikationsausdruck*. In Tabelle 1.1 ist die Synopsis der Unifikation verschiedener Kombinationen von Unifikationsausdrücken dargestellt.

Es gibt eine ganze Reihe von verschiedenen Verfahren, eine Übersicht findet sich in (Knight 1989). Der bedeutendste Unifikationsalgorithmus wurde von Herbrand (1930) entdeckt und von Robinson (1965) erneut gefunden. Dieser versucht, die Unifikationsargumente gemäß ihrer Syntax so „übereinanderzulegen“, dass eine Reihe von Differenzterm-Paaren entsteht. Diese Differenzen werden durch Substitution gemäß Tabelle 1.1 aufzulösen versucht. Trotz der großen Bedeutung wird hier darauf verzichtet, den *Herbrand-Robinson-Algorithmus* im Detail vorzuführen. Eine schöne Erklärung findet sich beispielsweise in (Bibel 1992). Albert et al. (1991) konnten zeigen, dass dieser Algorithmus im Mittel nur eine konstante Zahl von Schritten benötigt und sich unter diesem Aspekt als der bestmögliche erweist. Andererseits gibt es aber Termpaare, deren Unifikation exponentiellen Aufwand erfordert. Solche Fälle treten jedoch in der Praxis selten auf (siehe erneut (Albert et al. 1991)).

[8] Boolos und Jeffrey (1974) zeigen, wie zu einer gegebenen Turingmaschine eine endliche Satzmenge Δ und ein Satz H gefunden werden können, so dass $\Delta \vdash H$ genau dann gilt, wenn die Turingmaschine hält.

	Konstante	Variable	Funktion/ Prädikat
Konstante	Sind die Konstanten gleich: ERFOLG; sonst: FEHLER	Substitution {Variable/Konstante}	FEHLER
Variable	Substitution {Variable/Konstante}	Substitution {Variable/Variable}	Tritt die Variable nicht in der Funktion auf: Substitution {Variable/Funktion}; sonst: FEHLER
Funktion/ Prädikat	FEHLER	Tritt die Variable nicht in der Funktion auf: Substitution {Variable / Funktion}; sonst: FEHLER	Sind die beiden Funktionen/Prädikate gleich und ihre Argumente unifizierbar: Resultat der Unifizierung der Argumente; sonst: FEHLER

Tabelle 1.1: Unifikation verschiedener Kombinationen von Unifikationsausdrücken. Ergebnis der Unifikation kann „FEHLER“ (keine Unifikation möglich), direkter Erfolg (ohne Substitution) oder Erfolg mit Substitution sein.

Es gibt Unifikationsalgorithmen, deren Komplexitätsverhalten auf den ungünstigsten Unifikationsausdrücken – also im schlechtesten Fall[9] – deutlich besser ist. So haben Martelli und Montanari (1982) einen Algorithmus mit fast linearer Komplexität angegeben, dabei wird die Unifikation als Gleichungsproblem zwischen Differenztermen aufgefasst und spezielle Umformungsregeln spezifiziert.

Ein weiterer quasi-linearer Algorithmus ist von Huet (1987) vorgestellt worden. Der im worst-case schnellste Algorithmus wurde von Paterson und Wegman (1978) entwickelt. Er arbeitet auf gerichteten, azyklischen Graphen und implementiert aufwändige Graphen-Manipulationsverfahren.

1.3.2 Antwortextraktion mittels Resolution

Neben dem direkten Beweis eines Theorems zu gegebenen Axiomen ist ebenso die Beantwortung von Fragen an eine Wissensbasis eine Anwendung der automatischen Deduktion. Der Prozess der *Antwortextraktion* sammelt dabei die während den einzelnen Resolutionsschritten gewonnenen Variableninstanziierungen. Das Verfahren wurde von Green (1969) entwickelt und von Luckham und Nilsson (1971) analysiert und erweitert.

Bei der Antwortextraktion wird die Anwendung von Substitutionen während des Widerlegungsprozesses verfolgt. Dazu wird ein spezielles Antwortliteral $\texttt{Ans}(X_1, X_2, \ldots)$ an jede Klausel disjunktiv angehängt, die durch Umwandlung des zu beweisenden Theorems entsteht. Die Variablen von `Ans` stimmen dabei jeweils mit den Variablenvorkommen in der angehängten Klausel überein. Jede Resolution mit einer solchermaßen erweiterten Klausel hinterlässt daher ihre Spuren, nämlich ihre Substitutionen. Im Gegensatz zum vorgestellten Algorithmus 1.1 wird nun der Resolutionsprozess gestoppt, sobald eine Klausel hergeleitet wurde, die nur aus dem Antwortliteral besteht.

[9] engl. worst-case

Beispiel 1.3.1 (nach Nilsson (1980))
Gegeben sei das folgende Problem: „Wenn Fido immer dorthin geht, wohin auch John geht, und John in der Schule ist, wo ist dann Fido?“. Das führt zu den Sätzen

$$(\forall\ \texttt{X})\ [\texttt{AT(john,X)} \Rightarrow \texttt{AT(fido,X)}] \text{ und } \texttt{AT(john,school)}.$$

Die Frage $(\exists\ \texttt{X})\ \texttt{AT(fido,X)}$ wird wie üblich negiert, $(\forall\ \texttt{X})[\neg\texttt{AT(fido,X)}]$, und die entsprechende Klausel durch ein Antwortliteral disjunktiv erweitert. Man erhält $\neg\texttt{AT(fido,X)} \vee \texttt{Ans(X)}$. Die Resolutionswiderlegung ist in Abb. 1.2 gezeigt.

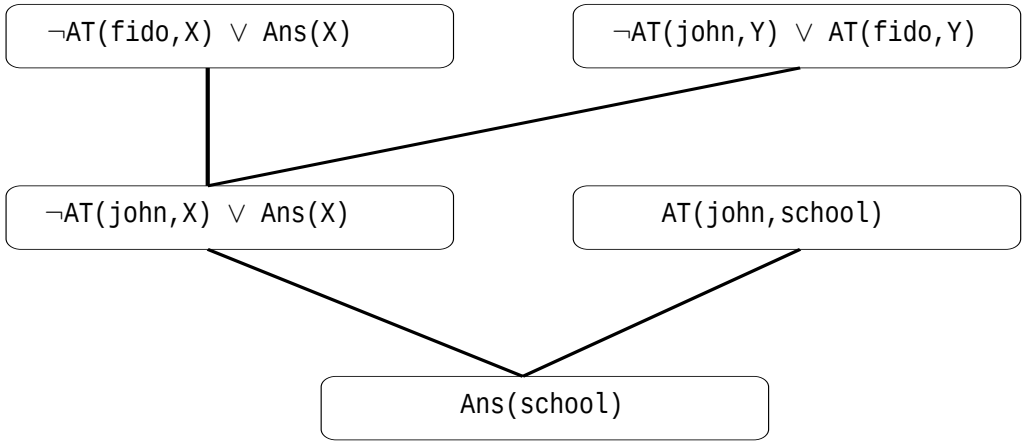


Abbildung 1.2: Widerlegungsbeweis mit Antwortextraktion zu Beispiel 1.3.1. Vergleiche Nilsson (1980).

Das vorstehende Beispiel zeigt, wie die Resolution zur Beantwortung solcher *Einsetzungsfragen* angewendet wird, man spricht auch von *Einsetzungsresolution*. Das Ergebnis der Einsetzungsresolution hängt von der jeweiligen Widerlegung ab. Im Allgemeinen können zu ein und derselben Frage verschiedene Widerlegungen existieren. Beim Prozess der Antwortextraktion wird eine der möglichen Antworten gegeben oder – wenn man den Prozess nicht nach der ersten Antwort stoppt – auch mehrere. Leider ist jedoch nicht feststellbar, ob zu einem bestimmten Zeitpunkt schon alle möglichen Antworten gefunden wurden. Grund ist die in Satz 1.2.2 angesprochene Unentscheidbarkeit der Prädikatenlogik 1. Ordnung.

Wird nur die Antwort auf eine einfache Wahr-/Falschfrage gesucht, so wird der normale Algorithmus zur Resolutionswiderlegung angewendet. Die Frage ersetzt in diesem Fall das zu beweisende Theorem; sie wird negiert, in Klauselform umgewandelt und der Menge der Axiome zugeschlagen. Eine leere Klausel konstituiert eine im Sinne der Frage positive Antwort.

1.3.3 Strategien zur Steuerung der Resolutionswiderlegung

Alle automatischen Beweiser sind im Prinzip Aufzählungsverfahren – sie verwenden einen Aufzählungsalgorithmus auf der Grundlage mindestens konsistenter Inferenzregeln. Im Unterschied zur mathematischen Logik besteht bei den maschinellen Beweisern aber auch der Zwang, den

Beweis auf eine effiziente Weise zu suchen. Der Algorithmus zur Resolutionswiderlegung (Algorithmus 1.1) sagt nichts über die Reihenfolge aus, in der Literale resolviert werden. Eine unkontrollierte Anwendung der Resolutionsregel hat den Nachteil, zahlreiche überflüssige Inferenzen zu erzeugen. So sind viele Inferenzen redundant, andere wiederum lenken den Resolutionsprozess in eine aussichtslose Richtung.

Praktikabel wird ein Deduktionssystem erst dann, wenn „schlechte“ Inferenzen weitgehend vermieden und „gute“ dagegen bevorzugt werden. Eine solche Auswahl setzt eigentlich bereichsspezifisches Problemwissen voraus, und das ist bei automatischen, nicht-interaktiven Beweisern, die allgemeine Probleme lösen, nicht vorhanden. Man kann sich jedoch einmal mehr auf rein syntaktische Gesichtspunkte zurückziehen und so problemunabhängig bleiben. Entsprechend schwach sind solche Verfahren; dennoch sind sie nach Bläsius und Bürckert (1992, Seite 79) ausschlaggebend dafür, dass automatische Deduktionssystem überhaupt funktionieren.

Dieses Kapitel stellt nun einige der wichtigsten Strategien vor, die im Rahmen dieser Arbeit von Interesse sind (eine ausführliche Diskussion erfolgt z. B. in (Chang und Lee 1973)). Die ersten drei sind *Restriktionsstrategien*[10], die letzte eine so genannte *Ordnungsstrategie*. Restriktionsstrategien verbieten einen Teil der möglichen Resolutionen von vorneherein und beschränken damit die Verzweigungsrate im Resolutionsgraphen. Dagegen beschränken Ordnungsstrategien diesen nicht, sie bestimmen jedoch die Reihenfolge, in der mögliche Inferenzen tatsächlich vollzogen werden.

Negative Resolution

Bei dieser Restriktionsstrategie wird verlangt, dass eine der an der Resolution beteiligten Klauseln (bei der binären Resolution sind dies zwei) eine Klausel ist, die ausnahmslos aus negativen Atomen aufgebaut ist. Diese Restriktion erhält (zusammen mit Resolution und Faktorisierung) die Widerlegungsvollständigkeit, siehe beispielsweise (Hofbauer und Kutsche 1991).

Unterstützungsmengen

Sehr häufig benutzt wird die so genannte *Unterstützungsmengen-Restriktion*, engl. set-of-support strategy (Wos et al. 1965). Zugrunde liegt die Annahme, dass die zur Diskussion stehenden Axiome nicht selbst widersprüchlich sind. Zur Widerlegung bedarf es einer Wechselwirkung – einer Resolution – zwischen Axiomenmenge einerseits und der Menge der Klauseln, die von der negierten Behauptung direkt oder indirekt abstammen, andererseits.

Diese Restriktion verbietet daher die Resolution zwischen Axiomen-Klauseln untereinander. Man sagt, diese haben keine *Unterstützung*. Unterstützung hingegen haben die Klauseln, die aus der Umwandlung des negierten zu beweisenden Theorems entstanden sind. Eine Resolution ist nur dann erlaubt, wenn mindestens eine der zu resolvierenden Klauseln Unterstützung hat. Falls eine Resolution möglich ist, wird die Unterstützung auf den Resolventen fortgeschrieben. Resolutionen finden also nur zwischen Klauseln statt, wenn sich mindestens eine „Eltern-Abstammungslinie“ auf die Behauptung zurückführen lässt.

[10] auch *Resolutionsstrategien* genannt, engl. resolution strategies

Unit-Resolution

Die *Unit-Resolution*, auch unitäre Resolution genannt, erlaubt eine Resolution genau dann, wenn mindestens eine der beteiligten Klauseln eine unitäre Klausel ist (siehe Def. 1.2.9). Diese Strategie ist, obwohl nicht widerlegungsvollständig, sehr populär, da sie stark zielorientiert arbeitet, nämlich in Richtung immer kürzer werdender Klauseln, bis schließlich die leere Klausel, ein Widerspruch, abgeleitet wird.

Stufensättigungsstrategie

Die Ordnungsstrategie der *Stufensättigung*, engl. level saturation, ordnet jeder Klausel eine *Ableitungslänge* zu. Die Ausgangsklauseln haben Länge Null. Die Länge einer Resolvente ist um eins größer als die größte der Elternklauseln. Bei Faktoren wird die Länge der jeweiligen Elternklausel ohne Änderung übertragen. Bei der Strategie der Stufensättigung werden Resolventen einer Stufe $k+1$ erst dann abgeleitet, wenn alle Resolventen der Stufe k erzeugt wurden. Das Verfahren ordnet also die möglichen Inferenzen nach der Ableitungslänge der erzeugten Klausel.

Diese Strategie ist für die in dieser Arbeit vorgestellte Form der Resolution *erschöpfend*, d. h. für jede überhaupt im Suchraum vorkommende Klausel ist sichergestellt, dass sie nach endlich vielen Schritten abgeleitet wird (falls das System nicht schon vorher durch erfolgreiche Widerlegung terminiert). Diese Strategie findet darüber hinaus den kürzesten Beweis, sofern es denn einen gibt.

Nachteile der Restriktionsstrategien

Restriktionsstrategien sind trivialerweise korrekt. Darüber hinaus erhalten die ersten beiden der hier vorgestellten Restriktionen die Widerlegungsvollständigkeit. Das ist jedoch – wie bei der unitären Resolution gesehen – nicht im Allgemeinen der Fall. Auch muss bei der Kombination verschiedener Strategien sehr behutsam vorgegangen werden, will man den Widerlegungssuchbaum nicht zu stark einschränken. Schon elementare Kombinationen können den Verlust der Widerlegungsvollständigkeit zur Folge haben. Einige Kombinationen und deren Eigenschaften finden sich in (Hofbauer und Kutsche 1991).

Durch die Anwendung der Restriktionen wird die Verzweigungsrate zum Teil wesentlich reduziert. Allerdings kann sich die Länge eines Beweises gegenüber einem unbeschränkten System dadurch auch verlängern, so dass nach Bläsius und Bürckert (1992) der Gesamtnutzen zweifelhaft ist. Dies geschieht immer dann, wenn die Herleitung einer Klausel verhindert wird, die für einen kurzen Beweis erforderlich ist. Eine Diskussion hierzu findet sich beispielsweise in (Görtz et al. 2000). Auch ist die tatsächliche Einschränkung des Suchraums in der Theorie schwer zu bestimmen. Ein erster theoretisch angelegter Vergleich verschiedener Restriktionsstrategien unter diesem Aspekt findet sich in Plaisted und Zhu (1999). Insgesamt sind die heute bekannten Resultate gerade in Hinblick auf die Prädikatenlogik nicht erschöpfend.

1.4 Aussagekraft und Komplexität der Prädikatenlogik 1. Ordnung

> *[...] more or less anything can be stated in first-order logic if you work hard enough at it.*
>
> Russell und Norvig (2003, Seite 272)

Ein wesentlicher Unterschied zu der entscheidbaren Aussagenlogik liegt in der Möglichkeit, neben der Referenzierung von Objekten eines Weltausschnitts auch Aussagen *über* diese zu machen und solche Aussagen zu benennen. Die Verwendung von Quantoren gibt die Flexibilität, einerseits über alle Objekte des Diskurses Aussagen zu machen, ohne sie einzeln aufzuzählen, oder andererseits die Existenz eines Objekts mit angegebenen Eigenschaften anzunehmen.

Durch Verwendung der logischen Operatoren können dann aus einfachen Sätzen komplexe aufgebaut werden. Dabei muss aber nicht die Wahrheit oder Falschheit der zugrunde liegenden Sätze (Konstituentensätze) spezifiziert werden. Beispielsweise macht der folgende Satz keine Aussage darüber, ob X größer als Y ist oder nicht:

$$\text{Größer}(X,Y) \vee \text{Kleiner}(X,Y).$$

Die auf diese Weise implementierte Konzeptionalisierung kann auf vielfältige Modelle (Kombinationen aus Domänen und Interpretationen) und Variablenzuordnungen angewendet und der Wahrheitsgehalt bezüglich dieses Kontextes überprüft werden.

Ein weiteres Merkmal der Prädikatenlogik ist (und das unterscheidet sie von der Aussagenlogik), dass Eigenschaften über unendliche Domänen ausdrückt werden können. Das erst ermöglicht ihre Anwendung bei der adäquaten Beschreibung mathematischer Begrifflichkeiten. Nicht verwechselt werden darf das aber mit einer Überabzählbarkeit der Modelle. Hier liegt eine Beschränkung, denn nicht alle Begriffe können durch eine rekursiv aufzählbare PL1-Axiomatisierung gefasst werden. Ein Beispiel dafür ist die Struktur der reellen Zahlen. Der Übergang von den abzählbaren zu den überabzählbaren Modellen wird erst ermöglicht, wenn gleichzeitig über Objekte und Prädikate quantifiziert werden kann, was in der Prädikatenlogik 1. Ordnung nicht hinreichend gelingt. Andererseits kann man jedoch interessante Teileigenschaften in PL1 formulieren und beweisen (Bläsius und Bürckert 1992).

Es gibt in der PL1 keine Aussagen, die ausschließlich auf Überabzählbares zutreffen. Das Theorem von Löwenheim und Skolem belegt, dass überabzählbare Strukturen nicht eindeutig charakterisiert werden können (Grosche et al. 1995). Mit anderen Worten können für Sätze zu überabzählbaren Strukturen auch immer unerwünschte Interpretationen gefunden werden.

Es existieren aber auch abzählbare Strukturen, die nicht in PL1 endlich-axiomatisierbar sind, wenn ihre Eindeutigkeit bis auf Isomorphie gefordert ist. Ein bekanntes Beispiel sind die natürlichen Zahlen. Sie werden beschrieben durch die *Peano-Arithmetik* (Peano 1889). Der in der Prädikatenlogik 1. Ordnung beschreibare Teil, genannt Robinson Arithmetik (Robinson 1950), definiert eine Basiskonstante 0 und eine injektive Nachfolgerfunktion $s(X)$, so dass jede natürliche Zahl außer der 0 ein Nachfolger von genau einer anderen ist. Ein Modell dieser Axiome ist das intendierte Modell, nämlich die Menge der natürlichen Zahlen $\mathbb{N}$. Es gibt aber andere, nicht-intendierte Modelle; die Ableitung von Theoremen ist in solchen Fällen nur beschränkt sinnvoll. In Abhängigkeit vom

Interpretationsbereich haben Prädikatsausdrücke verschiedenen Sinn. Um die nicht-angestrebten Modelle auszuschließen, ist das nicht mehr in PL1 formulierbare *Induktionsaxiom* erforderlich:

$$\forall P\ P(0) \wedge (\forall X\ P(X) \Rightarrow P(s(X))\ \Longrightarrow\ \forall X\ P(X), \text{ wobei } P \text{ ein Prädikat bezeichnet.}$$

Induktionsbeweise sind gerade die Beweise, die von diesem Axiom abhängen und können im Allgemeinen nicht in dieser Logik geführt werden. Im Speziellen kann die Induktion jedoch durch ein *Axiomenschema*, eine Art Vorlage für Axiome, bezeichnet werden. Dieses Vorgehen führt zwar zu unendlich vielen Axiomen, die Axiomenmenge ist jedoch rekursiv, es ist also durch Zusatzmechanismen möglich, zu entscheiden, ob ein gegebener Satz Axiom der Peano-Arithmetik ist (vgl. dazu (Goubault-Larrecq und Mackie 1997)).[11]

Die 1. Prädikatenlogik kann nicht nur diese abgeschwächte Form der Arithmetik ausdrücken, sondern sogar die Mengenlehre, engl. set theory, z. B. durch die Zermelo- und Frænkel-Axiome (eine Diskussion dieser Axiome sowie die sich aus ihnen ergebenden Konsequenzen findet sich in (Johnstone 1992) oder (Nerode und Shore 1997)). Die Theorie von Mengen ist von immenser Bedeutung, denn darauf aufbauend kann ein großer Teil der Mathematik axiomatisiert werden. Insbesondere können damit Listen (siehe (Russell und Norvig 2003)) und andere, komplexere Datenstrukturen axiomatisiert werden. Das ermöglicht die Programmverifikation durch automatische Deduktion im Prädikatenkalkül 1. Ordnung.

Nach einem Satz von Per Lindström (Lindström 1969) ist PL1 überhaupt die ausdrucksstärkste vollständige Logik. Jede Erweiterung wie beispielsweise die schon angesprochene Möglichkeit, über Funktions- und Prädikatsbezeichner zu quantifizieren, ist entweder nicht wirklich ausdrucksstärker und daher in PL1 simulierbar, oder es gibt kein vollständiges Kalkül mehr dafür (vgl. auch (Ebbinghaus et al. 1978)). Im letzten Fall ist nicht jeder allgemeingültige Satz durch endlich viele Anwendungen von Inferenzregeln aus gegebenen logischen Axiomen ableitbar.

Umgekehrt kann man sich fragen, ob denn erzwungene Einschränkungen der Prädikatenlogik 1. Ordnung Vorteile bringt. Brachman und Levesque (1984) sowie Levesque (1986) schlagen vor, die Ausdruckskraft von Repräsentationssprachen so stark zu begrenzen, dass ein polynomielles Antwortverhalten bei relevanten Schlussfolgerungsproblemen garantiert werden kann. Ein Beispiel ist *2SAT*, die Sprache der aussagenlogischen, konjunktiv verbundenen Klauseln, in der jede Klausel genau zwei Literale enthält. Das Erfüllbarkeitsproblem in *2SAT* ist effizient, d. h. mit polynomiellen Aufwand zu lösen.

Die Forderung wird jedoch heftig kritisiert, siehe beispielsweise (Doyle und Patil 1991). Hauptkritikpunkt ist, dass die Einschränkung der Ausdruckskraft einer Beschreibungssprache, die nötig ist, um Polynomialität in der Berechnung zu erlangen, diese so ausdrucksschwach macht, dass sie kaum noch verwendbar ist.

Betrachtet man die Ausdruckskraft von PL1 im Vergleich mit konventionellen Programmiersprachen, so ist festzustellen, dass jedes Programm einer Turingmaschine durch eine PL1-Satzmenge beschreibbar ist (dieses Resultat ergibt sich direkt aus der Betrachtung von Turing (1936)). Damit ist gezeigt, dass die Prädikatenlogik 1.Ordnung mindestens dieselbe Aussagekraft hat wie jede gebräuchliche Programmier-Hochsprache.

[11] Im Übrigen gilt nach Gödels erstem Unvollständigkeitssatz ganz allgemein, dass die Arithmetik nicht entscheidbar ist (Gödel 1931). Es gibt daher wahre arithmetische Sätze, die in keiner Logik bewiesen werden können.

1.5 Andere Logiken

Es gibt neben der klassischen Logik eine Reihe anderer formaler Systeme, die jeweils unterschiedliche Konzepte abdecken. Eine Auswahl wichtiger Ansätze ist die folgende:

- Intuistische Logik
- Temporale Logik
- Modale Logik
- Lineare Logik

Die klassische Logik, die die Prädikatenlogik beinhaltet, kann als eine Art von „Alltags-Logik“ angesehen werden. Während hier der Wahrheitsbegriff eine zentrale Rolle einnimmt, geht es bei der intuistischen Logik mehr um Beweise als um Wahrheit. Das Beispiel `A or (not A)` zeigt den Unterschied. Klassisch ist dies eine Tautologie, also immer wahr. In intuistischer Logik kann aber kein Beweis gefunden werden, da es im Allgemeinen keinen Beweis für `A` oder `not A` gibt.

Temporale Logik arbeitet den Begriff der Zeit in einen Beweis ein und ermöglicht, dass Sätze zu bestimmten Zeitpunkten wahr sind. Das ermöglicht die Beschreibung zeitabhängiger Szenarien, beispielsweise die Verifikation von Datentransferprotokollen (Herrmann und Krumm 2000). Modale Logik verallgemeinert dieses Konzept und erlaubt eine sich verändernde Interpretation. Die Wahrheit des Prädikats `verheiratet(Michael, Andrea)` hängt dabei von dem gegebenen Kontext ab. Geht es um heute oder morgen, um die Realität oder eine Traumwelt? Geht es um eine Heirat im kirchlichen Sinne? Bei der linearen Logik geht es hingegen um den Begriff der Ressourcen. Es müssen z. B. alle gegebenen Voraussetzungen in einem Beweis tatsächlich benutzt werden.

Im klassischen Bereich ist die umfassendste Erweiterung der PL1 die *Prädikatenlogik höherer Ordnung*. Ab der zweiten Ordnung ist es erlaubt, über endlichstellige Prädikate zu quantifizieren; diese Prädikate können dabei selbst wieder Argumente anderer Prädikate sein. Das erlaubt nun die Spezifizierung von Sätzen, die zu keiner Satzmenge der PL1 mehr logisch äquivalent sind. Beispiele sind das Induktionsaxiom oder das Axiom von der oberen Grenze bei den reellen Zahlen.

Diese Ausdrucksstärke wird dadurch bezahlt, dass die logische Implikation nicht mehr syntaktisch beschreibbar ist (Gödelscher Unvollständigkeitssatz, (Gödel 1931)). Mit anderen Worten gibt es Sätze, deren Allgemeingültigkeit mit keinem denkbaren Kalkül mehr nachgewiesen werden kann.

1.6 Hinweise zur Literatur

Diese Einführung in die Prädikatenlogik 1. Ordnung basiert auf den Büchern von Genesereth und Nilsson (1987), Hofbauer und Kutsche (1991), Fitting (1996) sowie Russell und Norvig (2003). Empfohlen werden außerdem die guten deutschsprachigen Einführungen von Winston (1987) und Bläsius und Bürckert (1992).

1.7 Zusammenfassung

Die Kodierung deklarativen Wissens erfolgt in einem mehrstufigen Prozess. Erster Schritt ist die Erstellung einer geeigneten Konzeptionalisierung des Anwendungsgebietes. Diese wird dann in syntaktische Konstrukte umgesetzt durch Festlegung von Objekt-, Funktions- und Prädikatsbezeichnern. Im Anschluss können *atomare Sätze* und schließlich *Sätze* formuliert werden. Die Semantik reichert diese Konstrukte mit Bedeutung und letztlich mit Wahrheitsgehalt an. Durch Angabe einer intendierten Interpretation sowie der Domäne wird ein Modell geschaffen. Modelle verknüpfen so Logiksymbole mit Welten. Dieses erlaubt, *wahre* Sätze zu formulieren, denn im Normalfall ist das Ziel, dass die niedergeschriebenen Sätze auch in der zu beschreibenden Wirklichkeit wahr sind. Andersherum kann die Konzeptionalisierung und ihre Umsetzung getestet werden an der Erfüllbarkeit der Satzmenge in einem Modell (Frage nach der Existenz eines Modells, Erfüllbarkeitsproblem). Eine weitere Frage ist, ob die gefundenen Modelle das intendierte Modell einschließen. Es ist die Aufgabe der Datenbasis, Modelle, die inkonsistent mit der intendierten Welt sind, auszuschließen (Russell und Norvig 2003).

Das Prädikatenkalkül besteht aus[12]

1. einer Beschreibung, wie syntaktisch korrekte Ausdrücke konstruiert werden können (wohlgeformte Sätze),
2. einem Axiomensatz (möglicherweise unendlich groß), wobei jedes dieser Axiome auch ein wohlgeformter Satz ist,
3. eine (endliche) Menge von Inferenzregeln, die es erlauben, aus den Axiomen und schon bewiesenen Theoremen neue Theoreme abzuleiten, d. h. zu beweisen.

In einem Kalkül verknüpfen Beweise Axiome mit Theoremen (Konsequenzen). Die Ableitung von Theoremen ist eine rein syntaktische Aktivität; sie geschieht durch Inferenzprozesse (Beweisprozeduren). Eine der wichtigsten Inferenzprozesse stützt sich auf die Resolutions-Inferenzregel, die konsistent und widerlegungsvollständig ist. Dieser Inferenzprozess beinhaltet einen Variablen-Substitutionsprozess (Unifikation) und erfordert Axiome im Klauselformat. Theoreme werden durch Widerlegung bewiesen. Suchstrategien helfen bei der Größenreduktion der entstehenden Suchbäume, allerdings kann keine resolvierende Strategie die kombinatorische Explosion des Suchraums verhindern. Im Prädikatenkalkül ist Beweisen ein mit exponentieller Komplexität behafteter Prozess. Ebenso ist dieses Kalkül nur semientscheidbar, da der Prozess auf eine Version des Halteproblems zurückgeführt werden kann. Die Semientscheidbarkeit dient andererseits als formale Rechtfertigung aller Deduktionssysteme: Ist ein Theorem wahr, kann dies auch in endlich vielen Schritten nachgewiesen werden. Ist ein Theorem aber falsch, so muss der Theorembeweiser nicht terminieren. In diesem Falle bleibt die Antwort offen. Das Prädikatenkalkül hat aber mindestens die Ausdrucksstärke traditioneller Programmiersprachen.

[12] Diese Kurzfassung ist (Wikipedia 2004) entnommen.

Kapitel 2

Grundlagen Künstlicher Chemien

> *Der Ursprung des Lebens vor vier Milliarden Jahren ist noch immer rätselhaft.*
>
> Frankfurter Allgemeine Zeitung (FAZ 2004)

Seit jeher ist die Suche nach dem Ursprung des Lebens eine der treibenden Kräfte der Wissenschaft. Über Jahrhunderte hinweg wurden Daten gesammelt, die die Diversität des Lebens dokumentieren. Hauptsächlich die Wissenschaft der Biologie dokumentierte und kategorisierte Daseinsformen und Prozesse des Lebens. Informationstheoretische Untersuchungen wurden dabei jedoch lange Zeit vernachlässigt. Die Artificial Life-Forschung[1] versucht diese Lücke zu füllen. Christopher Langton formulierte zum ersten Mal 1987 auf einem Workshop am Los Alamos National Lab die Artificial Life-Idee in dieser Form. Dieser junge Forschungszweig abstrahiert von einzelnen, spezifischen Lebensvorgängen. Das Ziel ist die Extraktion der Gesetzmäßigkeiten des Lebens und charakteristischer Erscheinungen. Dazu werden Herangehensweisen und Untersuchungsmethoden verschiedener Forschungsrichtungen zu einem interdisziplinären Denkansatz kombiniert.

Die Arbeitshypothese der Artificial Life-Forschung ist die Modellierbarkeit biotischer Phänomene durch komplexe Systeme vieler miteinander interagierender Komponenten. Dabei ist zentral der Begriff der *Emergenz*. Globale Systemeigenschaften werden aus lokalen Interaktionen von Teilsystemen gefolgert. Diese Interaktionen können sehr einfachen Regeln folgen. Entscheidend ist, dass das Verhalten des Gesamtsystems nicht aus der Analyse der Teilkomponenten vorhergesagt werden kann, es emergiert vielmehr aus ihrem Zusammenspiel. Es gilt hier der aristotelische Satz, dass das Ganze mehr als die Summe seiner Teile sei. Ein Gesamtsystem hat bestimmte Eigenschaften, nicht weil die Komponenten diese Eigenschaft aufweisen, sondern aufgrund ihrer *Organisation* und ihrer Funktion innerhalb des Verbundes.

Künstliche Chemien (KC) sind ein Teilgebiet der Artificial Life-Forschung. Sie modellieren künstliche Systeme, die der Chemie ähneln. Die Abstraktion von konkreten physikalischen Randbedingungen realer chemischer Systeme und natürlichen molekularen Prozessen ermöglicht eine

[1] Die deutsche Übersetzung „Künstliches Leben“ ist auch im deutschsprachigem Raum weniger verbreitet und wird daher im Folgenden nicht weiter verwendet.

Beschreibung dynamischer Vorgänge auf einer allgemeineren Ebene. Unterschiedliche Systeme werden dadurch vergleichbar. Biologie wird dabei als „Transformation von Organisationen“ betrachtet. Das globale Verhalten setzt sich zusammen aus lokal interagierenden Komponenten.

Eine Künstliche Chemie ist bestimmt durch ihre *Stoffe*, deren *Reaktionen* und einen *Reaktionsraum*. Ein solches mehr oder weniger abstraktes Reaktionssystem ist geeignet, Organisation, Selbstorganisation, Autopoiesis und allgemeine Strukturbildung zu untersuchen. Künstliche Chemien ermöglichen das Studium grundsätzlicher Mechanismen des Lebens, dabei können Fragen zum Ursprung und zur Evolution von Organisationen ergründet werden. Damit haben sie das Potenzial, Anworten auf bisher ungelöste Fragen zu geben.

Solche ungelösten Fragen verbinden sich beispielsweise mit der von Darwin postulierten Evolutionstheorie. Das Prinzip von Veränderung und Auslese, entdeckt durch Beobachtung des Verhaltens von Populationen in einer begrenzten Umwelt, erklärt die Evolution nur teilweise. Wie entstehen die Voraussetzungen, also die Entitäten, die variiert und selektiert werden? Im Rahmen Künstlicher Chemien können die dynamischen Vorgänge solcher komplexen Systeme unter Zuhilfenahme der Abstraktion vom natürlichen Vorbild simuliert werden.

2.1 Überblick

Die Chemie beschreibt das Zusammenwirken von Materie auf einem niedrigen Niveau. Allerdings sind die grundlegenden Mechanismen skalierbar und so auf andere dynamische Systeme gewinnbringend übertragbar. Hofbauer und Sigmund (1988) zeigen sogar eine Übertragung auf soziale Systeme. Eine Künstliche Chemie definiert ein abstraktes Reaktionssystem, in dem Stoffe miteinander reagieren und damit die Konzentrationen der Stoffe im System verändern oder neue Stoffe bilden.

2.1.1 Definition

Eine allumfassende Definition Künstlicher Chemien ist angesichts der in der Literatur zu findenen Vielfalt schwierig. Wie abstrakt sollte eine Künstliche Chemie sein? Wie nah an der Chemie darf sie angesiedelt sein? Eine allgemeine Definition ist die nach Dittrich (2001):

Definition 2.1.1 (Künstliche Chemie)
Eine Künstliche Chemie (KC) ist ein von Menschen konstruiertes System, das der Chemie ähnelt.

Die Allgemeinheit dieser Definition stellt sicher, dass keine relevanten Arbeiten ausgeschlossen werden.

Nach Dittrich et al. (2001) kann eine Künstliche Chemie im Normalfall definiert werden durch das Tripel (S,R,A). S bezeichnet dabei die Menge aller im System vorkommenden Moleküle. Die Reaktionen zwischen diesen Molekülen werden definiert durch die Menge R der Reaktionsregeln. Sie legt das Interaktionsverhalten der Moleküle fest. Schließlich legt ein Algorithmus A die Dynamik des Systems fest durch Wahl einer Reaktortopologie und durch Festlegung der Reihenfolge, mit der die Reaktionsregeln aus R angewandt werden. Sowohl die Moleküle als auch die Reaktionsregeln können explizit oder implizit definiert werden.

2.1.2 Abstraktionsniveau

Gemäß des Abstraktionsgrads werden *analoge* und *abstrakte* Künstliche Chemien unterschieden. Gibt es einen Isomorphismus zwischen Molekülen des künstlichen Systems und der Chemie oder einen Isomorphismus zwischen den Reaktionen, so handelt es sich um eine analoge KC. Diese sind schon verwandt zu Modellen der *Computational Chemistry*, die eine möglichst genaue Modellierung der Chemie anstreben. Eine abstrakte KC dagegen zieht ihren Nutzen hauptsächlich aus der qualitativen Übertragung von Reaktionsregeln. Das verwendete Abstraktionslevel wird bestimmt durch den mit der Modellierung verfolgten Zweck.

2.1.3 Motivation – Anwendungsgebiete

Grundmotivation vieler Forschungsaktivitäten mit Künstlicher Chemie ist, abstrakte Modelle zu entwickeln, die die grundsätzlichen Eigenschaften von Systemen abbilden, die auf biologische oder biochemische Mechanismen zurückgreifen. Unter Zuhilfenahme der Abstraktion vom natürlichen Vorbild der molekularen Prozesse werden durch Simulation die dynamischen Vorgänge in komplexen Systemen untersucht. Gesucht werden bei dieser KC-Forschung vor allem Strukturen, die in der Lage sind, sich und andere im Laufe eines Experiments entstehende Strukturen zu verändern oder zu erhalten, sowie komplexere Strukturen auszubilden. Im Mittelpunkt steht das Studium allgemeiner Phänomene: (präbiotische) Evolution, Selbstorganisation, Metabolismus, Entstehung von Wettbewerb und Kooperation usw. Die schöpferische Leistungskraft (im Englischen ist hier der Begriff *constructiveness* gebräuchlich) der Künstlichen Chemien erlaubt die einfache Konstruktion dynamischer Systeme. Die Arbeiten auf diesem Gebiet sind sehr vielfältig. Eine detaillierte Übersicht gibt (Dittrich et al. 2001). Beispiele für die Modellierung von Teilgebieten (Subsystemen) komplexer biologischer Systeme sind allgemeine Zellwachstumsmodelle (Astor und Adami 1998; Furusawa und Kaneko 2000; Ono und Ikegami 2000) und die Modellierung eines Ökosystems (Suzuki et al. 2000).

Neben biologischen und strukturellen Aspekten sind im Bereich der Modellbildung auch andere Systeme von Interesse, die prinzipielle Ähnlichkeiten mit chemischen Systemen aufweisen, so zum Beispiel soziale Systeme (Hofbauer und Sigmund 1988; Szuba und Straš 1997) oder parallele Rechenprozesse (Berry und Boudol 1992). Ihnen gemeinsam ist die Metapher der kollidierenden

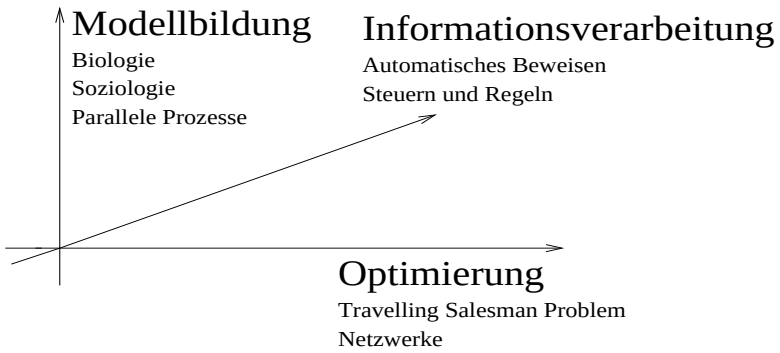


Abbildung 2.1: Schematische Darstellung der unterschiedlichen Einsatzgebiete von KC.

Moleküle. Abbildung 2.1 gibt einen Überblick über das gesamte Spektrum der KC-Forschung (nach Skusa et al. (2000)).

In den Bereichen *Optimierung* und *Informationsverarbeitung* sollen die Erkenntnisse über die Funktionsweise dezentraler und paralleler Systeme praktisch nutzbar gemacht werden. Ausgenutzt wird dabei, dass viele (bio)chemische Prozesse in der Natur als Berechnungen interpretiert werden können. Die Verwendung Künstlicher Chemien führt dabei über eine rein theoretische Betrachtung hinaus. Die Metapher der KC kann hier Basis für die Entwicklung von Algorithmen und großen Datenstrukturen sein. Emergente und dezentrale Steuerung durch lokale, parallel ablaufende Prozesse sind übertragbare Konzepte. Beispiele für praktische Anwendungen: Robotersteuerung (Ziegler und Banzhaf 2000), Problemoptimierung (Kanada und Hirokawa 1994), Sortierung von Zahlen (Banzhaf et al. 1996) und Astrobiologie (Centler et al. 2003). Bei der Optimierung geht es zumeist um kombinatorische Probleme. Dieser Teil der KC-Forschung wird im Nachfolgenden die *anwendungsorientierte* Seite der KC genannt. Im Gegensatz dazu stehen die anfangs erwähnten grundlagentheoretischen Fragestellungen.

2.2 Komponenten einer Künstlichen Chemie

Die folgenden Unterkapitel stellen nacheinander die Bestandteile einer KC vor: Molekülmenge, Reaktionsregelmenge und Steueralgorithmus. Letzterer bestimmt das Verhalten des Reaktors – und damit die Dynamik des Systems.

2.2.1 Festlegung der Molekülmenge

In einer durch (S,R,A) spezifizierten Künstlichen Chemie legt die Menge S die in ihr gültigen Moleküle fest, die auch Objekte genannt werden. Diese Moleküle reagieren entsprechend der durch R gegebenen Reaktionsvorschriften. Die Festlegung der Moleküle kann explizit oder implizit erfolgen:

Explizite Festlegung Bei dieser Festlegung werden die Moleküle explizit aufgezählt. Die Definition $S = \{a, b, 0, 1, +\}$ ist ein Beispiel.

Implizite Festlegung Implizit gegeben ist die Molekülmenge, wenn eine Konstruktionsanleitung vorliegt, also eine Beschreibung, wie Objekte konstruiert werden. Die Definition $S = \{0,1\}^*$ ist eine solche implizite Festlegung und bezeichnet die Menge aller Binärstrings. Implizite Moleküldefinitionen sind häufig die Grundlage für eine konstruktive dynamische KC (siehe Unterkapitel 2.4), die die Bildung neuer Molekültypen ermöglicht. Die Moleküle stehen dabei nicht a priori fest. Typische Beispiele sind Zeichenketten (Bagley und Farmer 1992; Kauffman 1993; McCaskill et al. 1994), Binärstrings (McCaskill 1988; Banzhaf 1993) und Lambda-Ausdrücke (Fontana 1992).

In Künstlichen Chemien finden sich eine große Vielfalt von Moleküldefinitionen. Komplexere Moleküle können in hierarchisch oder objektorientierter Weise konstruiert werden (Bersini 2000). Die Molekülrepräsentation wird oft auch als *Struktur* bezeichnet.

2.2.2 Spezifikation der Reaktionen

Die Menge der Reaktionsregeln R, kurz Regelmenge, beschreibt die Interaktionen zwischen Objekten. Es hängt von der Struktur der Objekte und der Zielsetzung der KC ab, welche Regeln in die Menge R aufgenommen werden, denn sie legen die Reaktionen zwischen kollidierenden Objekten fest. Eine Regel $r \in R$ kann in vereinfachter chemischer Notation geschrieben werden in der Form

$$s_1 + s_2 + \cdots + s_n \rightarrow s'_1 + s'_2 + \cdots + s'_m,\ s_i \in S. \tag{2.1}$$

Eine Anwendung der Regel r ersetzt die n Objekte auf der linken Seite, *Edukte* genannt, durch die m Objekte der rechten Seite, die *Produkte*. n gibt die Ordnung der Reaktion an.[2] Im Allgemeinen wird eine Reaktion als symmetrisch bezüglich ihrer Edukte angenommen, d. h. die Reaktion $s_1 + s_2$ liefert die gleichen Produkte wie die Reaktion $s_2 + s_1$. Führt eine Kombination von Molekülen zu keiner Reaktion, ist also keine Reaktionsregel in R vorhanden, gemäß derer eine Ersetzung durch Produkte stattfinden kann, so wird die Reaktion als *elastisch* bezeichnet. Andernfalls heißt eine Reaktion *produktiv*. Wiederum kann zwischen expliziter und impliziter Festlegung der Regeln unterschieden werden:

Explizite Reaktionsdefinitionen Eine explizite Interaktionsdefinition ist unabhängig von der Struktur der Moleküle. Explizit wird jede Reaktion zwischen Molekülen in R aufgezählt. Alle in der KC vorkommenden Molekültypen und deren Interaktionen sind von vorneherein bekannt. Die Anzahl der Molekültypen ändert sich nicht.

Implizite Reaktionsdefinitionen Sind die Reaktionen impliziert definiert, so beziehen sie sich auf die Struktur der Moleküle. Die Zahl der Moleküle darf unendlich sein, denn Reaktionsprodukte müssen nicht in einer Aufzählung explizit genannt werden, sondern ergeben sich durch die Anwendung einer Reaktionvorschrift auf den Edukten. Damit sind implizite Reaktionsdefinitionen geeignet für die Bildung einer konstruktiven dynamischen KC (siehe Unterkapitel 2.4). Implizite Reaktionsvorschriften sind beispielsweise durch das Lambda-Kalkül (Fontana 1992), Matrixmultiplikation (Banzhaf 1993) oder endliche Automaten (Dittrich 1995) gegeben.

2.2.3 Steueralgorithmus – Dynamik

Der Algorithmus A bestimmt, wann welche Objekte miteinander reagieren, wie mit dem Reaktionsprodukt verfahren wird, und welche Randbedingungen eingehalten werden müssen. Die Zusammenfassung der Moleküle wird *Population*, *Reaktor*, oder auch *Suppe* genannt. Denkbar sind Simulationen eines gut gerührten Reaktors ohne Topologie (Bagley und Farmer 1992; Fontana 1992; Banzhaf 1993), mit euklidischer Topologie (Varela et al. 1974), mit einem zweidimensionalem Gitter (Banzhaf et al. 1999) oder mit einem kontinuierlichen dreidimensionalen Raum (Zauner und Conrad 1998). Reaktormodelle können geschlossen sein oder über einen Zufluss verfügen. Sie haben entweder konstante Größe oder weisen im Verlauf der Zeit eine variierende Zahl von Molekülen auf.

[2] Gegenüber der Chemie bedeutet das eine Vereinfachung.

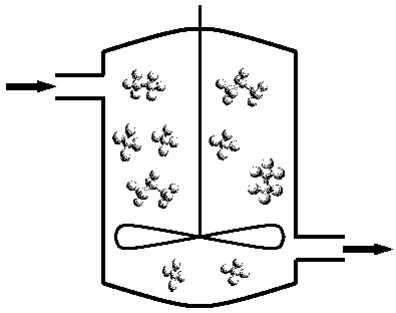


Abbildung 2.2: Allgemeine Form eines gut gerührten Reaktors. Ein Rührer sorgt für gleichmäßige Durchmischung. Neue Moleküle entstehen durch Reaktion oder treten durch Zufluss in den Reaktor ein. Durch den Abfluss von Molekülen kann die Reaktorgröße konstant gehalten werden.

Das Verhalten des Reaktors ergibt sich aus dem Steueralgorithmus. Er bestimmt die Dynamik der Künstlichen Chemie. Der Aufbau des Algorithmus hängt ab von der Repräsentation der Molekülpopulation. Ein viel verwendetes Modell ist das des *gut gerührten Reaktors*, auch Rührkessel genannt (Abb. 2.2). Ein Rührer sorgt für gleichmäßige Durchmischung des Inhalts, so dass beliebige Moleküle jederzeit kollidieren können. Es existiert so keine räumliche Struktur und die Population kann repräsentiert werden explizit als Multimenge[3] oder implizit als Konzentrationsvektor.

2.3 Dynamische Simulation

Es gibt zwei grundsätzliche Herangehensweisen, die durch A festgelegte Dynamik eines Reaktors zu modellieren. Einerseits kann jedes Molekül einzeln betrachtet werden (explizite Modellierung), andererseits können alle Moleküle gleichen Typs zusammengefasst repräsentiert werden durch die zu diesem Molekültyp gehörende Konzentration (implizite Modellierung).

2.3.1 Stochastische Molekülkollisionen

Dieser Ansatz simuliert jede Kollision explizit. Eine festgelegte Anzahl von Molekülen wird dem Reaktor entnommen. Gibt es eine Regel r aus R, deren linke Seite vollständig und korrekt mit den gewählten Molekülen in Übereinstimmung gebracht werden kann, so findet eine Reaktion statt. Die durch die rechte Seite von r spezifizierten Moleküle bilden die Reaktionsprodukte, die dem Reaktor wieder zugeführt werden. Durch erneute Entnahme von Molekülen und versuchter Regelanwendung ergibt sich ein Zyklus, der im Folgenden *Reaktionszyklus* genannt wird. Der einmalige Durchlauf dieses Zyklus führt zu einer Erhöhung der Simulationszeit um das Inkrement $\frac{1}{n}$, wobei n die Zahl der Moleküle im Reaktor, also die Populationsgröße, bezeichnet. Nach n simulierten Kollisionen erhöht sich so die Simulationszeit um einen Zeitschritt, gemeinhin wird

[3] Eine Multimenge ist eine Menge, in der Elemente mehrfach auftreten können.

von einer *Generation* gesprochen. Da die Kollisionen gezählt werden, ist es für diese Zeitrechnung unerheblich, ob diese Kollisionen produktiv oder elastisch sind, also Reaktionen nach sich ziehen oder nicht. Die Definition einer Zeiteinheit als Durchführung von n Kollisionen hat den Vorteil, dass Reaktoren unterschiedlicher Größe in gleicher Zeit gleiche Entwicklungsfortschritte erzielen können.

Die dem Reaktor zugrunde liegende Topologie bestimmt den Ziehungsvorgang der Moleküle. Beim gut gerührten Reaktor werden die potentiellen Edukte zufällig gezogen. Dieser Basisalgorithmus kann erweitert werden durch die Modellierung eines Zu- und Abflusses.

2.3.2 Kontinuierliche Differenzial- bzw. diskrete Differenzgleichungen

Oft erfolgt die Simulation der Dynamik chemischer Systeme durch gewöhnliche Differenzialgleichungen (Dittrich et al. 2001). Die implizite Modellierung beruht auf den Konzentrationen verschiedener Molekültypen. Die Differenzialgleichungen beschreiben die Entwicklung dieser Konzentrationen. Es bezeichne x_i die Konzentration des Molekültyps s_i und N die Zahl unterschiedlicher Molekültypen. Dann kann eine Reaktion $r \in R$ geschrieben werden als

$$r : a_1 s_1 + a_2 s_2 + \ldots + a_N s_N \rightarrow b_1 s_1 + b_2 s_2 + \ldots + b_N s_N. \tag{2.2}$$

Die Koeffizienten a_i und b_i bezeichnen die stöchiometrischen Faktoren. Ist ein Molekültyp s_i nicht Edukt der Reaktion, gilt $a_i = 0$, ist s_i kein Produkt, gilt $b_i = 0$.

Nach dem Massenwirkungsgesetz von Guldberg und Waage (1879) (siehe auch Koudriavtsev et al. (2001)) gilt in idealer Situation, dass die Reaktionsrate proportional zum Produkt der Massen der reagierenden Substanzen ist. Nach Dittrich et al. (2001) wird bei elementaren Wechselwirkungen durch die Reaktion r die folgende Konzentrationsänderung des Moleküls s_i herbeigeführt:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = (b_i - a_i) \prod_{j=1}^{N} x_j^{a_j}, i = 1, \ldots, N. \tag{2.3}$$

Um den Einfluss jeder Reaktion aus R auf Molekültyp s_i zu beschreiben, wird über alle $r \in R$ summiert:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_{r \in R} (b_i^r - a_i^r) \prod_{j=1}^{N} x_j^{a_j^r}, i = 1, \ldots, N. \tag{2.4}$$

Diese Gleichungen sind ein kontinuierliches Modell für eine diskrete Situation. Eine Modellierung der Dynamik durch Differenzialgleichungen stellt eine Approximation für Systeme dar, in denen große Mengen jeder Molekülspezies vorhanden sind. Abbildung 2.3 zeigt anhand einer einfachen KC die Approximation der DGL-Simulation im Vergleich zu expliziten Molekülkollisionen in Reaktoren verschiedener Größe.

Der Berechnungsaufwand eines Differenzialgleichungssystems hängt von der Zahl seiner Gleichungen ab, hier von der Zahl N der verschiedenen Molekülsorten. Eine Dynamiksimulation durch Differenzialgleichungen ist daher nur für Künstliche Chemien mit einer relativ kleinen Zahl verschiedener Molekültypen effizient.

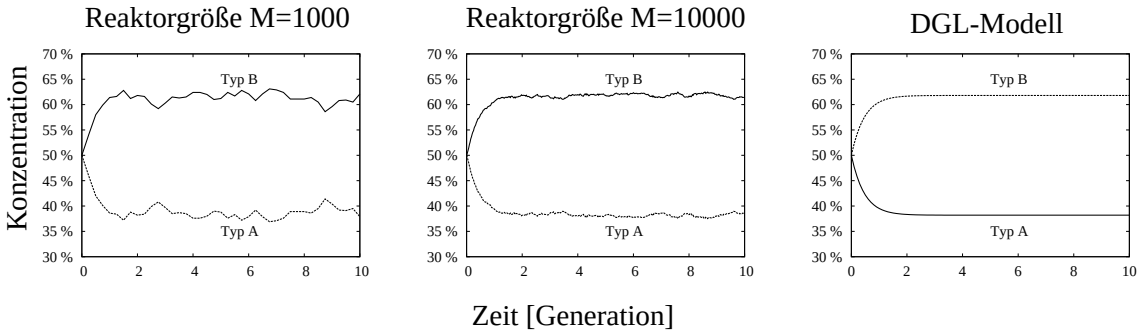


Abbildung 2.3: Vergleich der expliziten Simulation mit der durch ein Differenzialgleichungssystem. Beispiel ist eine einfache KC mit 2 Molekültypen A und B. Die Reaktionsregelmenge R besteht aus 4 Regeln: $R = \{A+A \rightarrow A+A+B,\ A+B \rightarrow A+B+B,\ B+A \rightarrow B+A+B,\ B+B \rightarrow B+B+A\}$. *Links:* Explizite Simulation in *RESAC* mit 1000 Molekülen. *Mitte:* Explizite Simulation in *RESAC* mit 10000 Elementen. *Rechts:* Analytische Lösung des DGL-Systems.

2.3.3 Metadynamik

Farmer et al. (1986) und Bagley und Farmer (1992) stellen einen Ansatz vor, in dem auch auf Differenzialgleichungen zurückgegriffen wird. Allerdings ändern sich diese Gleichungen mit der Zeit. Nur die Molekültypen, die oberhalb eines Schwellwerts liegen, werden durch Gleichungen berücksichtigt. Mit Änderung der Molekülkonzentrationen werden neue Gleichungen ins System aufgenommen bzw. Gleichungen entfernt.

2.4 Konstruktive dynamische Systeme

Nach Meinung mehrerer Autoren gibt es eine Klasse von Systemen, die mit klassischen Methoden nur unzureichend beschrieben, erklärt oder vorhergesagt werden können (Kampis 1991; Rosen 1991; Fontana 1992): die *konstruktiven dynamischen Systeme*. Diese Klasse von Systemen ist charakterisiert durch die Fähigkeit, aus sich selbst heraus neue Komponenten bilden zu können. Die neu gebildeten Komponenten nehmen Einfluss auf die Dynamik des Systems und ändern diese möglicherweise (Fontana 1992). In konventionellen nicht-konstruktiven dynamischen Systemen sind dagegen von Beginn an alle beteiligten Komponenten und Interaktionen bekannt.

Künstliche Chemien sind besonders gut geeignet, konstruktive dynamische Systeme zu modellieren, da Objekte und deren Interaktionen frei festgelegt werden. In der Simulation können dann die dynamischen Prozesse, die Prinzipien und das Potenzial des Paradigmas der Konstruierbarkeit studiert werden. Es ist zu beachten, dass KC-Systeme nicht per se konstruktiv dynamisch sind:

Definition 2.4.1 (Konstruktive dyn. KC-Systeme – Def. nach Banzhaf (2004))
Ein durch die Künstliche Chemie $K = (S, R, A)$ gegebenes dynamisches System DS wird konstruktiv genannt, wenn seine Komplexität bezüglich eines Komplexitätsmaßes C, das die Zahl der aktuell im System vorhandenen Komponenten und die Zahl der bis dahin durchgeführten Interaktionen berücksichtigt, mit der Zeit ansteigt. Für $C(DS(K))$ gilt dann: $C(t_2) > C(t_1)$ mit $t_2 > t_1$.

Werden neue Systemkomponenten zufällig konstruiert, spricht man von einem *schwach konstruktiven System*, entstehen die neuen Komponenten dagegen durch Interaktion anderer Komponenten, so wird das System als *stark konstruktiv* bezeichnet. Diese Klassifizierung geht auf Fontana et al. (1994) zurück.

2.5 Selektion, Variation und Emergenz

Das neo-Darwinistische Paradigma legt der Evolution die grundsätzlichen Mechanismen der Replikation, der Selektion und schließlich der Variation zugrunde. Die Replikation stellt sicher, dass Information konserviert werden kann. In einem System ohne Informationsbewahrung ist keine nachhaltige Entwicklung zu erzielen. Wettbewerb entsteht durch Selektion: In einem Umfeld begrenzter Ressourcen haben Entitäten mit hoher Qualität eine größere Reproduktionswahrscheinlichkeit. Es ist nicht immer klar, worin das Qualitätsmaß, die so genannte *Fitness* besteht; üblicherweise wird darunter ein Wettbewerbsvorteil verstanden, eine größere Überlebenswahrscheinlichkeit. In der präbiotischen Evolution jedoch ist nicht erkennbar, worin der Selektionsvorteil besteht, der dazu geführt haben könnte, dass sich einzelne Moleküle zu größeren funktionellen Einheiten wie Makromolekülen oder Membranstrukturen organisiert haben (Fontana 1992). Kauffman (1993) weist darauf hin, dass der Neo-Darwinismus nicht erklärt, wie die reproduzierenden Einheiten selbst entstanden sind, also wie die Voraussetzungen für diese Evolutionstheorie geschaffen wurden. Fontana und Buss (1994) drückten es in Anlehnung an die bekannte These vom „Überleben des Stärkeren“, engl. survival of the fittest, so aus, dass die „Ankunft des Stärkeren“ nicht erklärt werden kann. Dieser Umstand gilt dabei für jede fundamentale Entwicklungsstufe.

Die Künstlichen Chemien kommen ganz ohne die Mechanismen Selektion und Variation aus. Trotzdem können Prozesse simuliert werden, in denen sich manche Stoffe gegenüber anderen durchsetzen. Dominanz, Selbstorganisation und Aussterben können sich als Folge vieler Wiederholungen von festgelegten Reaktionen zwischen zufällig ausgewählten Molekülen ergeben. Fontana und Buss (1994) zeigen die Entstehung von Zyklen sich gegenseitig erzeugender Moleküle, Banzhaf (1994b) beschreibt den Aufbau von Komplexität in einem System. Künstliche Chemien sind daher geeignet, Fragen aus der Evolutionstheorie zu untersuchen. Sie tun dies, ohne Selektions- und Variationsmechanismen einsetzen zu müssen. Die Diskussion, ob der erste Schritt in der Entwicklung von Leben Replikatoren oder sich selbst unterhaltende autokatalytische Verbände waren, gehört so beispielsweise zu den mit einer KC untersuchten Aspekten (Eigen und Schuster 1977; Farmer et al. 1986). Ein anderes Beispiel ist die Untersuchung von Bagley und Farmer (1992) über die Entstehung eines Metabolismus aus zufälligen Anfangsbedingungen.

Die angesprochene Betrachtung sehr vieler lokaler Einzelreaktionen, die wieder und wieder durchgeführt werden und so neue Stoffe hervorbringen bzw. bestehende verändern, führt zum Begriff der *Emergenz* bzw. des *emergenten Verhaltens*.

Definition 2.5.1 (Emergentes Verhalten – Def. nach Steels (1994))
Ein Verhalten ist emergent, wenn es nur durch Grundaussagen definiert werden kann, die nicht gebraucht werden, um das Verhalten der konstituierenden Einzelkomponenten zu beschreiben.

Emergenz bezeichnet die Entstehung neuer Strukturen oder Eigenschaften aus dem Zusammenwirken der Elemente eines Systems.

2.6 Analyse Künstlicher Chemien

Die Simulation Künstlicher Chemien liefert eine große Menge von Daten. Experimente mit 10^6 Molekülen und 10^{10} Kollisionen sind nicht untypisch. Die während eines Experiments gesammelte Datenflut erfordert besondere Analysemethoden. *Mikroskopische* Methoden betrachten einzelne Moleküle oder Reaktionen. Den Betrachtungen liegt in den meisten Fällen der Begriff der *Generation* zugrunde. Eine Generation ist definiert als ein Zeitraum, in dem genau M Kollisionen stattfinden. M bezeichnet dabei die Zahl der Moleküle im Reaktor. Die *Molekülkonzentration* eines Molekültyps errechnet sich aus dem Verhältnis von Anzahl der Moleküle dieses Typs zu M. Konzentrationsentwicklungen spezieller Stoffe und ihre Einbindung in Reaktionsnetzwerke können auf dieser Mikroebene verfolgt werden.

Konzentrationen und Integration in Reaktionspfade ändern sich im Allgemeinen dynamisch. Diese Daten sind daher typischerweise über den gesamten Simulationszeitraum zu erheben. Es kommt hinzu, dass oft nicht a priori bekannt ist, welche Objekte in der nachfolgenden Analyse von besonderer Wichtigkeit sein werden. Interessant sind gerade Strukturen, die erst im Verlaufe des Experiments entstehen. Es kann im Normalfall nicht vorhergesagt werden, *wann* und *welche* Strukturen entstehen, oft nicht einmal, von welcher Art die neu entwickelten komplexen Strukturen sein werden. Das macht Analysemethoden auf einer *Makroebene* erforderlich. Weitere Methoden lassen sich zwischen Mikro- und Makroebene einstufen.

Weit verbreitete makroskopische Maßzahlen sind *Produktivität* und *Diversität*. Die Diversität bezeichnet den Anteil unterschiedlicher Objekte in der Population. Die Produktivität beziffert den Anteil produktiver Reaktionen bezogen auf die Gesamtzahl der Kollisionen. Im Gegensatz zu den produktiven Reaktionen erzeugen elastische Reaktionen kein Reaktionsprodukt.

Zur Bestimmung der Produktivität, die die Wahrscheinlichkeit angibt, dass eine Kollision von Molekülen eine (produktive) Reaktion nach sich zieht, werden in einer laufenden Simulation innerhalb eines Zeitfensters alle nicht-elastischen Reaktionen gezählt. Die erhaltene Anzahl wird zu der Größe des betrachteten Zeitfensters ins Verhältnis gesetzt. Üblicherweise wird eine Generation betrachtet, also die Zahl der nach n Kollisionen durchgeführten Reaktionen durch n geteilt, wobei n die Größe einer nicht-leeren Population bezeichnet. Die Diversität setzt die Zahl unterschiedlicher Molekültypen zur Größe n einer nicht-leeren Population ins Verhältnis.

Sprünge oder Unregelmäßigkeiten in Diversitäts- oder Produktivitätskurven kennzeichnen besondere Entwicklungsphasen des Experiments. Sie können beispielsweise die Evolution neuer Strukturen und Selbstorganisation anzeigen (Skusa et al. 2000).

Teil II

Das System *RESAC*

Kapitel 3

Aufbau von *RESAC*

Die resolutionsbasierte Künstliche Chemie *RESAC* führt die Konzepte zweier Wissensgebiete zusammen, des Prädikatenkalküls einerseits und der Künstlichen Chemien andererseits. Abbildung 3.1 verdeutlicht das Ineinandergreifen von Prädikatenkalkül und Künstlicher Chemie. *RESAC* ist wie andere Künstliche Chemien üblicherweise auch (siehe Kap. 2) definiert durch das Tupel (A, S, R). Der Algorithmus A legt das dynamische Verhalten des Systems durch präzise Definition des Reaktors fest. Die Moleküle innerhalb des Reaktors sind gegeben durch die Objektpopulation S. Die Reaktionsregelmenge R bestimmt, auf welche Weise kollidierende Moleküle reagieren bzw. ob sie überhaupt miteinander reagieren. Die Struktur der Objekte aus S, also die Molekülstruktur, wird in *RESAC* durch Elemente des Prädikatenkalküls festgelegt. Ähnliches gilt für die Aufstellung des einzigen verwendeten Reaktionsschemas. Hier wird zur Implementierung des Interaktionsverhaltens eine Inferenzregel des Kalküls auf *RESAC* transferiert.

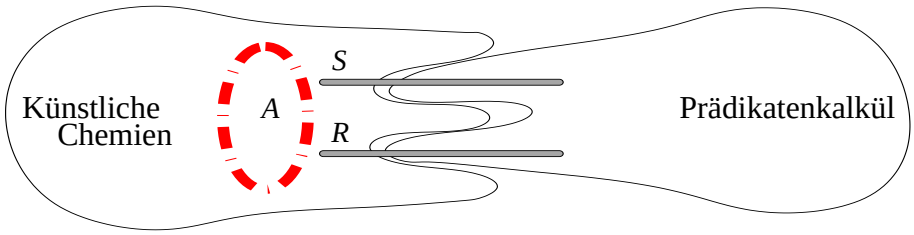


Abbildung 3.1: Schematisches Zusammenwirken der beiden Komponenten Künstliche Chemie – hier definiert durch Tupel (A, S, R) – und Prädikatenkalkül innerhalb des Systems *RESAC*. Die Steuerung, gegeben durch Algorithmus A, besteht aus einem Reaktor und einem Reaktionszyklus. Sie greift so auf Konzepte der Künstlichen Chemien zurück. Die Molekülstruktur und das Interaktionsschema (bezeichnet mit R) entstammen dem Prädikatenkalkül. Die Menge der Moleküle ist bezeichnet mit S.

3.1 Konstruktion der Künstlichen Chemie *RESAC*

Wie schon angesprochen ist es günstig, die Beschreibung einer Künstlichen Chemie in drei Teile zu separieren: (1) Moleküle, (2) Reaktionsregelmenge und (3) Dynamik. Im ersten Teil wird die Menge aller gültigen Moleküle definiert, die in der Chemie auftreten können. Teil zwei beschreibt, was passiert, wenn sich Moleküle nahe kommen oder treffen. Zu diesem Zweck werden im Allgemeinen ein oder mehrere *Reaktionsschemata* spezifiziert, die alle Reaktionen zwischen Edukten beschreiben. Schließlich wird durch den letzten Teil festgelegt, wie der Reaktor konstruiert ist und wie sich Moleküle in ihm verhalten.

Bevor nun die Komponenten formal eingeführt werden, sei zum Überblick noch auf die Analogien zwischen Prädikatenkalkül und *RESAC* hingewiesen. Abbildung 3.2 stellt die Konzepte gegenüber.

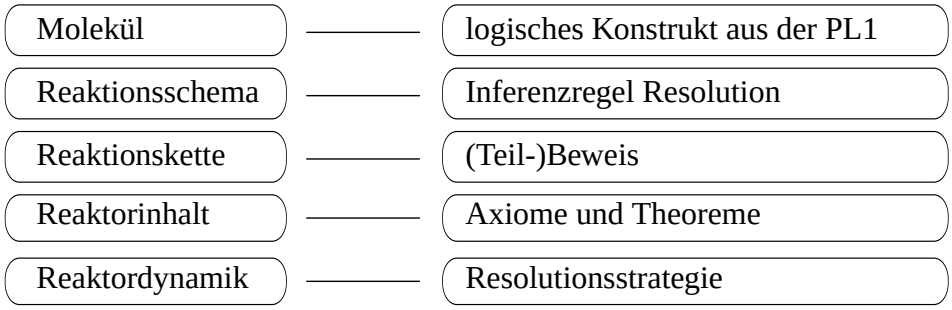


Abbildung 3.2: Analogie zwischen *RESAC* (links) und Prädikatenkalkül (rechts). Jede Reaktion zwischen Molekülen entspricht der Anwendung der Resolutions-Inferenzregel. Reaktionsketten entsprechen einem Beweis des Produkts. Der Reaktorinhalt beschreibt durch die Menge der abgeleiteten logischen Konstrukte Axiome und darauf basierende Theoreme. Die durch einen Algorithmus vorgegebene Dynamik des Reaktors ist vergleichbar mit Resolutionsstrategien (siehe dazu Kap. 1.3.3).

3.1.1 Moleküle

In *RESAC* tragen die Moleküle logische Information. Sie bezeichnen Klauseln gemäß Def. 1.2.8 über einer nach Def. 1.1.2 durch $\mathcal{L}(\mathcal{P},\mathcal{F},O)$ gegebenen Prädikatenlogik 1. Ordnung. Da in *RESAC* keine andere Logik Verwendung findet, wird aus Gründen der Übersichtlichkeit auf eine formale Parametrisierung der Moleküle mit den Mengen $\mathcal{P},\mathcal{F},O$ der Relations-, Funktions- und Objektbezeichnern verzichtet. Die PL1 liegt – sofern nicht abweichend spezifiziert – den folgenden Definitionen implizit zugrunde. Die Menge S der Moleküle ist dann gegeben durch

$$S = \{\langle\Phi\rangle_K \mid \Phi \in \mathcal{L}(\mathcal{P},\mathcal{F},O)\}.$$

Dabei bezeichnet $\langle\Phi\rangle_K$ die Wandlung des PL1-Satzes Φ in Klauselform (siehe Kapitel 1.2.2, *Klauselform*). Die Menge der gültigen und stabilen Moleküle umfasst so die Menge aller PL1-Klauseln.

3.1.2 Reaktionsregelmenge

Die Reaktionsregelmenge besteht nur aus einem einzigen, implizit definierten Reaktionsschema in zwei Variationen. Beiden liegt zugrunde die Inferenzregel der binären Resolution gemäß Def. 1.2.12, im Folgenden bezeichnet mit Res_2. Das Schema arbeitet ausschließlich auf der Struktur der Edukte. Da zugleich die Moleküle nach Konstruktion logische Konstrukte desselben Kalküls sind, ist dieses Schema wohldefiniert.

Die Anwendung der Resolution Res_2 kann auf allen Molekülen versucht werden, allerdings ist sie nur erfolgreich, wenn ein komplementäres und unifizierbares Literalpaar auffindbar ist. Nur in diesem Fall resultiert aus der Anwendung von Res_2 eine Resolvente, die das Reaktionsprodukt bildet. Andernfalls findet keine Reaktion statt, gekennzeichnet durch das Symbol $\perp$. Der Zugewinn an Molekülen durch Reaktionen, der so genannte *Reaktionszugewinn*, wird wie folgt durch die Funktion R_Δ beschrieben:

$$R_\Delta : S \times S \longrightarrow S \cup \{\perp\}$$

$$\text{mit } (\phi,\psi) \longmapsto \begin{cases} \perp & Res_2 \text{ nicht anwendbar auf } (\phi,\psi), \\ Res_2(\phi,\psi) & sonst. \end{cases} \tag{3.1}$$

Gilt $R_\Delta(\phi,\psi) = \perp$ für Reaktionspartner ϕ und ψ, so wird die Reaktion als *elastisch* bezeichnet und die beteiligten Moleküle bleiben unverändert. Andernfalls spricht man von einer *produktiven* Reaktion. Vereinfachend soll $R_{\Delta,\text{productive}}$ geschrieben werden, wenn eine produktive Reaktion vorausgesetzt und auf die Fallunterscheidung in der Definition von R_Δ verzichtet werden kann.

In *RESAC* wird das Reaktionsschema R in zwei Varianten angeboten, die sich in der Weise unterscheiden, wie im Falle einer produktiven Reaktion mit dem Reaktionsprodukt verfahren wird. Die erste Variante, R_{Edukt} genannt, ersetzt teilweise die Edukte der Reaktion, die zweite Variante, R_{Free} genannt, verdrängt ein beliebiges Molekül des Reaktors. Was mit diesem geschieht, wird durch die Festlegung auf eine bestimmte Reaktordynamik entschieden, z. B. fließt es aus dem Reaktor ab.

$$R_{\text{Edukt}} : S \times S \longrightarrow \mathcal{P}(S)$$

$$\text{mit } (\phi,\psi) \longmapsto \begin{cases} \{\phi,\psi\} & \text{wenn } R_\Delta(\phi,\psi) = \perp, \\ \{\rho, R_\Delta(\phi,\psi)\} & \text{sonst,wobei } \rho \in \{\phi,\psi\} \\ & \text{und } Prob(\rho = \phi) = \frac{1}{2}. \end{cases} \tag{3.2}$$

$$R_{\text{Free}} : S \times S \longrightarrow \mathcal{P}(S)$$

$$\text{mit } (\phi,\psi) \longmapsto \begin{cases} \{\phi,\psi\} & \text{wenn } R_\Delta(\phi,\psi) = \perp, \\ \{\phi,\psi,R_\Delta(\phi,\psi)\}, & \text{sonst.} \end{cases} \tag{3.3}$$

In der Ersetzungsvariante R_{Edukt} wird durch eine produktive Reaktion genau eines der Edukte gleichwahrscheinlich durch das Reaktionsprodukt $R_{\Delta,\text{productive}}$ ersetzt. In einer in der Chemie üblichen Notation kann vereinfachend geschrieben werden:

$$R_{\text{Edukt,productive}} : s_1 + s_2 \longrightarrow s_i + s_3,\ i \in \{1,2\},\ \text{wobei } s_3 = Res_2(s_1,s_2).$$

R_{Free} zeichnet sich dagegen durch eine Beibehaltung der Edukte auch im produktiven Fall aus:

$$R_{\text{Free,productive}} : s_1 + s_2 \longrightarrow s_1 + s_2 + s_3, \text{ wobei } s_3 = Res_2(s_1, s_2).$$

Die beiden autokatalytisch wirkenden Ersetzungsvarianten des Reaktionsschemas sprechen den Aspekt der *Ersetzungsmethodik* an, im Falle der durch R_{Edukt} charakterisierten Reaktionsregel soll im Folgenden von der Ersetzungsmethode (oder synonym: Ersetzungsschema) *Eduktersetzung* gesprochen werden. Die Ersetzungsvariante R_{Free} wird hingegen *freie* Ersetzung oder *Freiersetzung* genannt. Eine detaillierte Diskussion der Ersetzungsmethodik findet in Kap. 4 statt. Letztendlich ist genau eine dieser beiden Ersetzungsvarianten das einzige Element der Reaktionsregelmenge:

$$R_{RESAC} ::= \{R_{\text{Free}}\} \mid \{R_{\text{Edukt}}\}.$$

3.1.3 Dynamik

Der Reaktoraufbau und die Dynamik der ablaufenden Prozesse wird determiniert durch einen Algorithmus, im Folgenden durch *A* bezeichnet. Er legt die Topologie, Einbettung des Reaktors in seine Umwelt (Interaktion mit extern ablaufenden Prozessen), Filter-, Inspektions- und Entnahmeprozesse fest. Des Weiteren werden die Anfangsbedingungen spezifiziert, ebenso die Endkonditionen, die bestimmen, unter welchen Voraussetzungen ein Experiment terminiert wird. Auch die angesprochene Ersetzungsmethodik beeinflusst die Dynamik.

RESAC zugrunde liegt ein gut gerührter Reaktor, ein so genannter Rührkessel, engl. well-stirred reactor. Diese Topologie ist von vielen Autoren bei ihren Experimenten bzw. Modellen eingesetzt worden, einer der Ersten war vermutlich Eigen (1971) bei seinem vielbeachteten Konzept des Evolutionsreaktors. Ein Rührer wälzt dabei die Moleküle solchermaßen um, dass (idealisiert) davon ausgegangen werden kann, dass beliebige Moleküle quasi jederzeit miteinander reagieren können. Eine Diskussion dieses Konzepts findet sich in Unterkapitel 3.2. Es wird sich herausstellen, dass eine solche Annahme nicht unrealistisch ist. Das Konzept des gut gerührten Reaktors sorgt für eine Aufhebung der räumlichen Struktur. Der Reaktor kann modelliert werden durch eine Multimenge (siehe Unterkapitel 2.2.3). Zur Durchführung einer Kollision wird auf zwei zufällige Elemente dieser Menge zugegriffen und geprüft, ob eine Resolution zwischen diesen möglich ist. Ist das der Fall, findet eine Reaktion in Form der Anwendung der Resolutionsregel statt. Je nach implementierter Ersetzungsmethodik ersetzt das Produkt eines der Edukte oder ein beliebiges Element der Multimenge. Anhand der beiden Kriterien Reaktor-Abgeschlossenheit und Ersetzungsmethodik können in der Hauptsache vier verschiedene Reaktortypen unterschieden werden:

(1) geschlossene Reaktoren mit Eduktersetzung, (2) offene Reaktoren mit Eduktersetzung,

(3) geschlossene Reaktoren mit Freiersetzung, (4) offene Reaktoren mit Freiersetzung.

Das erste Unterscheidungskriterium spricht den Aspekt der Öffnung des Systems zur Interaktion mit der Außenwelt an. Dabei geht es in erster Linie darum, einen, mehrere oder auch keinen Zulauf zum Reaktor zu vereinbaren. In festgelegten, durch die Zuflussrate gesteuerten, Zeitabständen ersetzt ein Element der Menge *Startklauseln* ein Element der Multimenge. Über einen Ablauf verlassen ersetzte (d. h. überzählige) Moleküle den Reaktor. Das zweite Unterscheidungskriterium gibt an, ob mit dem Reaktionsschema der Eduktersetzung oder mit dem der freien Ersetzung gearbeitet werden soll. Aus der Kombination der Ausprägungen dieser beiden Kriterien ergeben sich die vier in Abbildung 3.3 gezeigten Reaktorvarianten.

Allgemeine Parametrisierung und Funktionsaufrufe der Reaktoralgorithmen

Startklauseln Den Reaktorvarianten gemein ist ihre Befüllung zu Anfang des Experiments. Sie wird bestimmt durch die Menge der so genannten *Startmoleküle*, synonym zu gebrauchen ist der Begriff *Startklauseln*, da Moleküle über S definiert sind.

Multiplizität Jedes Molekül aus der Menge *Startklauseln* wird mit der *Multiplizität* μ in die Reaktoren aufgenommen, d. h. im Reaktor befinden sich μ Kopien eines jeden Moleküls. Die Größe eines Reaktors $|M|$ ist demnach gegeben durch:

$$|M| = \mu * |\mathit{Startklauseln}|. \tag{3.4}$$

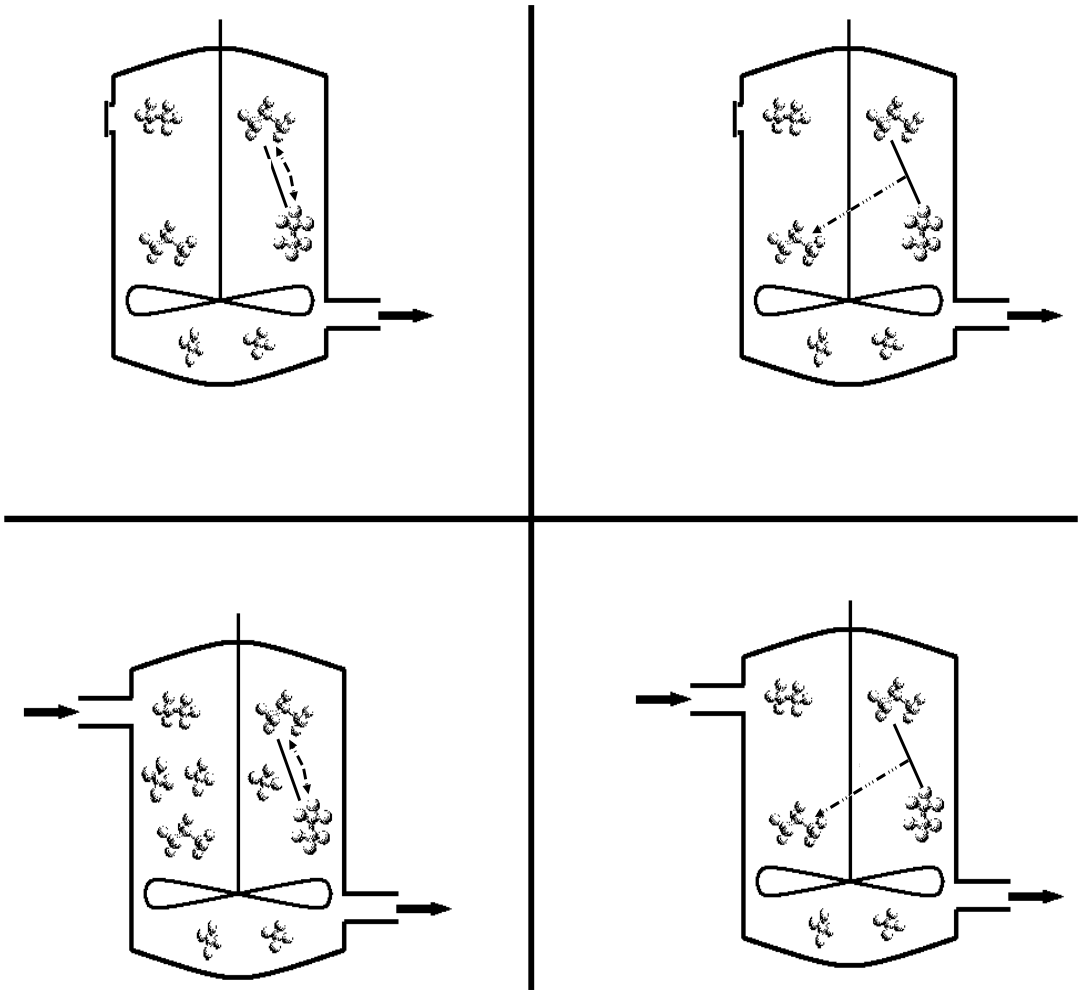


Abbildung 3.3: Die vier Reaktorbasistypen in *RESAC*. Bei dieser schematischen Darstellung wird abstrahiert von einem stetigen Zufluss an Energie und logischen Elementarteilchen, der allen Reaktortypen zugrunde liegt. *Oben:* geschlossene Reaktoren. *Unten:* offene Reaktoren. *Links:* Eduktersetzung. *Rechts:* freie Ersetzung.

Res$_2(\phi,\psi)$ Anwendung der binären Resolution auf die Moleküle ϕ und ψ. Dazu wird aus der Menge aller resolvierbaren Literalpaarungen zu ϕ,ψ (vgl. Def. 1.2.12) *zufällig* eine ausgewählt und zur Resolution herangezogen. Gibt es keine solche Paarung, ist der Funktionsrückgabewert das Symbol $\perp$. Andernfalls ergibt sich der Rückgabewert aus dem Resolutionsergebnis. Durch diese Funktionsvorschrift wird nicht-deterministisches Verhalten induziert. Zu beliebigen, aber fest vorgegebenen Molekülen *A* und *B* berechnet *Res*$_2(A,B)$ möglicherweise verschiedene Rückgabewerte.

terminate() Diese Funktion mit Wertebereich {`wahr`, `falsch`} bestimmt den Zeitpunkt der Terminierung eines Experiments. Sie verfügt dazu über sämtliche globale Information des Hauptalgorithmus. Ein Abbruch kann beispielsweise angeordnet werden durch Erfüllung von Voraussetzungen der folgenden Art: (1) Vorfinden eines bestimmten Moleküls im Reaktor, (2) Überschreitung bestimmter Konzentrationen vorher festgelegter Moleküle, (3) Unterschreitung eines vorgegebenen Niveaus durch ein Analysemaß wie z. B. der Produktivität, (4) Überschreitung eines *Zeitlimits*.

Zuflussrate i Dieser Parameter steuert die Zuflussrate in offenen Reaktoren. Das übernächste Unterkapitel liefert weitere Erklärung.

Geschlossene Reaktoren

Es folgen nun die Algorithmen für geschlossene Reaktoren beider Ersetzungsvarianten.

Algorithmus 3.1 (Algorithmus für geschlossene Reaktoren mit Eduktersetzung)
(* Eduktersetzung, geschlossener Reaktor *)

Input: (1) Menge $Startklauseln \subseteq S$ (2) Multiplizität μ

```
1  Fülle den Reaktor M zufällig mit μ Kopien jeder Klausel aus der Menge Startklauseln
2  ReaktionsNr := 0; GenerationsNr := 0
3  while ¬terminate() do
4        i ← randomInteger(1, μ * |Startklauseln|)
5        repeat j ← randomInteger(1, μ * |Startklauseln|)
6          until (j ≠ i)
7        k ← randomBoolean()
8        Erg ← Res2(M[i], M[j])                          Anwendung Resolution
9        if (Erg ≠ ⊥) then                               produktive Reaktion
10                       if (k = wahr)                   Eduktersetzung
11                         then M[i] ← Erg               1. Edukt wird ersetzt
12                          else M[j] ← Erg              2. Edukt wird ersetzt
13                       fi
14                 else                                  elastische Reaktion
15                       ; (* Moleküle an Positionen i, j bleiben unverändert *)
16       fi
17       ReaktionsNr ← ReaktionsNr + 1;
18       GenerationsNr ← GenerationsNr + 1/(μ*|Startklauseln|)
19 od
```

Algorithmus 3.2 (Algorithmus für geschlossene Reaktoren mit Freiersetzung)
(* Freiersetzung, geschlossener Reaktor *)

Input: (1) Menge $Startklauseln \subseteq S$ (2) Multiplizität μ

1 Fülle den Reaktor M zufällig mit μ Kopien jeder Klausel aus der Menge *Startklauseln*
2 $ReaktionsNr := 0$; $GenerationsNr := 0$
3 **while** $\neg terminate()$ **do**
4 $i \leftarrow randomInteger(1, \mu * |Startklauseln|)$
5 **repeat** $j \leftarrow randomInteger(1, \mu * |Startklauseln|)$
6 **until** $(j \neq i)$
7 $k \leftarrow randomInteger(1, \mu * |Startklauseln|)$
8 $Erg \leftarrow Res_2(M[i], M[j])$ Anwendung Resolution
9 **if** $(Erg \neq \perp)$ **then** produktive Reaktion
10 $M[k] \leftarrow Erg$ freie Ersetzung
11 **else** elastische Reaktion
12 ; (* alle Moleküle bleiben unverändert *)
13 **fi**
14 $ReaktionsNr \leftarrow ReaktionsNr + 1$
15 $GenerationsNr \leftarrow GenerationsNr + \frac{1}{\mu * |Startklauseln|}$
16 **od**

Reaktoren mit Zuflussmöglichkeiten

Die Algorithmen zur Modellierung offener Reaktoren mit Eduktersetzung beziehungsweise mit freier Ersetzung bauen auf den vorgestellten Algorithmen der jeweiligen Ersetzungsmethodik auf. Im Unterschied zu diesen verfügen sie über einen weiteren Eingabeparameter, der die *Zuflussrate* festlegt.

Algorithmus 3.3 (Algorithmus für Zuflussreaktoren)
(* Zuflussreaktoren, Frei- und Eduktersetzung *)

Input: (1) Menge $Startklauseln \subseteq S$ (2) Multiplizität μ (3) Zuflussrate $i > 0$

1 Fülle den Reaktor M zufällig mit μ Kopien jeder Klausel aus der Menge *Startklauseln*
2 $ReaktionsNr := 0$; $GenerationsNr := 0$
3 **while** $\neg terminate()$ **do**
4 $\ldots$ { Kopie der Zeilen 4 bis 18 des Algorithmus 3.1 Eduktersetzung; Kopie der Zeilen 4 bis 15 des Algorithmus 3.2 Freiersetzung }
5 **if** $(ReaktionsNr \mod round(\frac{1}{i}) = 0)$
6 **then** (* zufällige Startklausel einfließen lassen *)
7 $Zufluss \leftarrow Startklauseln(randomInteger(1, |Startklauseln|))$
8 $M[randomInteger(1, \mu * |Startklauseln|)] \leftarrow Zufluss$
9 **fi**
10 **od**

Die Rate i wird angegeben als rationale Zahl im Bereich zwischen 0 und 1. Eine Zuflussrate i, $0 < i \leq 1, i \in \mathbb{Q}$, bezeichnet dabei einen *Zufluss* (engl. inflow) in jeder $round(\frac{1}{i})$. Reaktion – ob nun produktiv oder elastisch. Für $i \to 0$ wird ein geschlossener Reaktor modelliert, der Algorithmus für Zuflussreaktoren geht dann in die Basisalgorithmen 3.1 und 3.2 über. Eine Zuflussrate von $i = 0.5$ führt nach jeder zweiten Reaktion zu einem Molekülzufluss. Ein stetiger Zufluss wird durch die Wahl einer Rate von $i = 1$ erreicht. Gebräuchlich ist auch die Sprechweise $(i * 100)$*%-Zufluss*.

Eine besondere Art des Zuflusses ist gegeben durch den so genannten *elastischen Zufluss*. Bei dieser Zuflussstrategie wird immer dann ein Molekül dem Reaktor zugeführt, wenn eine vorhergehende Kollision nicht zu einer produktiven Reaktion führte.

Der Zufluss besteht ausnahmslos aus Molekülen der Menge *Startklauseln*, dem ersten Übergabeparameter. Die Auswahl des zufließenden Moleküls aus dieser Menge erfolgt gleichverteilt. Das zufließende Molekül ersetzt ein beliebiges Molekül des Reaktors, welches sozusagen den Reaktor durch den Abfluss verlässt.

3.2 Diskussion des Reaktormodells

In diesem Kapitel soll das in Kapitel 3.1 eingeführte Reaktormodell des gut gerührten Reaktors vor dem Hintergrund realer Anordnungen in der Chemietechnik beleuchtet werden.

Nach Denbigh und Turner (1973) sind die vier wichtigsten Reaktortypen in der Chemie die folgenden:

1. Rohrreaktor,
2. Rührkessel für Satzbetrieb,
3. Rührkesselkaskade und
4. Wirbelbettreaktoren[1].

Der Rohrreaktor, auch Strömungsrohr genannt, verdankt seinen Namen der Tatsache, dass viele Reaktoren dieses Typs die Form eines Rohres haben. Im Allgemeinen versteht man darunter jede Art von kontinuierlich betriebenem Reaktor, durch den ein Materialstrom in einer bestimmmten Richtung fließt, die Stoffe treten dabei an einem Ende des Systems ein und am anderen aus. Der Rührkesselreaktor dagegen besteht aus einem Kessel mit intensiver Rührung, in den ein ständiger Strom von Reaktanden hineingeht und aus dem die (teilweise) abreagierten Stoffe kontinuierlich abgezogen werden. Dieser Reaktoraufbau dient als Vorlage des in *RESAC* modellierten gut gerührten Reaktors. Die in der Literatur gebräuchliche Abkürzung für Reaktoren dieses Typs ist *CSTR*, abgeleitet von der englischen Bezeichnung *continous flow stirred tank reactor*. Schaltet man mehrere Kessel in Serie, wobei die reagierenden Stoffe von einem Kessel zum nächsten fließen, so liegt eine Rührkesselkaskade vor.

Während beim Rohrreaktor für die grundlegende Berechnungsmethode ideale Kolbenströmung angenommen wird, geht man beim Rührkessel beziehungsweise der Rührkesselkaskade entsprechend von der (ebenfalls vereinfachenden) Annahme der vollständigen Vermischung in jedem

[1] Wirbelbettreaktoren sind in der vorliegenden Dissertation von untergeordnetem Interesse.

Kessel aus. Als Folge davon gilt, dass die Zusammensetzung des Gemisches am Ausgang des Reaktors identisch ist mit derjenigen an beliebigen anderen Stellen innerhalb des Kessels. Darin besteht der Hauptunterschied zum (idealen) Strömungsrohr, in dem keine Vermischung zwischen den in den Reaktor eintretenden Stoffen und den mehr oder weniger abreagierten Produkten stattfindet.

Die zur Vermischung des Reaktorinhalts solcher CSTR benötigte Zeit steht mit der „Zirkulationszeit“ der Reaktionsmischung im Tank im Zusammenhang. Wenn die Zirkulationszeit im Vergleich zur mittleren Verweilzeit kurz ist, kann man den Reaktor als ideal durchmischend ansehen, d. h. die Leistung des Reaktors wird nur unbedeutend von der aufgrund der Annahme idealer Durchmischung berechneten Leistung abweichen (Denbigh und Turner (1973)). Sofern die beteiligten Gemische nicht zu zäh sind, ist es nach (Hill 1977) nicht schwer, diese Bedingung in industriellen Aufbauten bzw. mit Ausrüstung industrieller Güte zu erfüllen.

Im Gegensatz zum Rohrreaktor bietet der Rührkessel einige Vorteile. Die gute Mischung bewirkt optimale Ausnutzung des Reaktorvolumens (es gibt annähernd keine Totvolumina). Er bietet darüber hinaus eine homogene Konzentrationsverteilung, im Reaktor selbst wie im Ablauf, als auch eine homogene Temperaturverteilung (gleiche Temperatur). Bedeutsam ist hier die Nichtexistenz punktueller örtlicher Überhitzungen. Mit Kühlschlangen (von innen) sowie mit Kühlaggregaten von außen ist eine Temperaturkontrolle möglich, hier muss nur die zeitverzögerte Antwort auf die veranlassten Kühlmaßnahmen beachtet werden.[2] Diese präzise Temperaturregelung ist in der realen Anwendung von zentraler Bedeutung, z. B. bei der Kontrolle der Bildung von unerwünschten Nebenprodukten.

Nachteilig wirkt sich beim Rührkessel der so genannte *Schlupf* aus. Infolge wirksamer Rührung kann sich ein in den Reaktor eintretendes Molekül schon im nächsten Moment an beinahe jeder beliebigen Stelle befinden. So kann es insbesondere zum Ablauf des Reaktors gelangen, ohne überhaupt in einer Reaktion involviert gewesen zu sein. Andererseits lassen sich auch solche Moleküle finden, die selbst nach längerer Zeit nicht ablaufen. Es gibt in jedem Kessel ein breites Spektrum von Verweilzeiten. Grundsätzlich gilt für den ideal durchmischten Rührkessel: Ein Molekül hat an jeder Stelle die gleiche Aufenthaltswahrscheinlichkeit.

Rührkessel sind in der angewandten Chemieindustrie weit verbreitet und gerade bei Flüssigphasereaktionen in der organischen Chemie sehr beliebt (Hill 1977). Beispiele sind die verschiedenen Polymerisationsreaktoren. Weiter weist Hill (1977) darauf hin, dass CSTR insbesondere bei Experimenten mit niedrigen Reaktionsgeschwindigkeiten Gewinn bringend eingesetzt werden.

Die Homogenität der Molekülkonzentrationen im Reaktorinnern, die zu einfachen aber trotzdem realistischen Modellierungsalgorithmen führt, das spontane Antwortverhalten auf Zuflussänderung, die einfache Reaktortemperaturkontrolle und der natürliche Skalierungsansatz der Rührkesselkaskade sind Argumente, die für die Wahl einer Rührkesselarchitektur in *RESAC* sprechen.

[2] Diese Kühlaggregate sind ihrerseits wieder gut vermischt, verfügen ebenso wie der Hauptreaktor über Zu- und Abfluss. Die durch die Reaktionen produzierte Energie kann so über die Reaktorwände aufgenommen und über den Kühlmittelabfluss abtransportiert werden.

3.3 Parallelisierung

In Künstlichen Chemien werden im Allgemeinen eine Vielzahl elementarer Interaktionen, d. h. Rechenschritte, ausgeführt. In *RESAC* besteht die elementare Interaktion darin, eine Kollision zwischen zwei Molekülen zu simulieren. Dazu werden die Moleküle als Klauseln der Prädikatenlogik aufgefasst und die Anwendungsmöglichkeit der Resolutionsregel geprüft. Gegebenenfalls wird diese zur Bestimmung des Reaktionsprodukts durchgeführt. Sind erst einmal die Reaktionspartner bestimmt, laufen alle dazu notwendigen Berechnungen lokal ab. Mit anderen Worten wird keinerlei Information zur Durchführung der Kollison gebraucht mit Ausnahme derjenigen, die in den Edukten enthalten ist. Eine Parallelisierung des Gesamtsystems ist daher effizient durchführbar.

Zwei verschiedene Herangehensweisen sind grundsätzlich denkbar:

(1) Parallelisierung der einzelnen Kollisionen.

Idee: Bildung von $\frac{n}{2}$ Molekülpaaren mit anschließender parallelen Berechnung der Kollision und gegebenenfalls der darauf folgenden Reaktion.

Realisierung: Thread-Programmierung. Molekülen, die gerade nicht in Reaktionen verwickelt sind, wird paarweise ein Thread zugeordnet und dieser gestartet.

Nachteile: Immenser Verwaltungsaufwand und Speicherbedarf (durch Stackallokation) durch die hohe Zahl gleichzeitig ablaufender Prozesse und deren unterschiedlicher Ausführungsgeschwindigkeit.

(2) Aufteilung des Reaktors in mehrere Bereiche mit parallelisierter Simulation der einzelnen Reaktorbereiche.

Idee: Aufbau von parallel arbeitenden, miteinander interagierenden Reaktoren.

Realisierung: Peer-to-Peer-Netzwerk mit zentraler Administration.

Im Rahmen dieser Arbeit wird der erste Ansatz nicht weiterverfolgt. Ziel ist nicht der Einsatz auf einem platzbeschränkten Parallelrechner mit festgelegter Prozessorzahl, beispielsweise einer PRAM-Architektur (siehe u. a. JaJa (1992), Grolmusz (1991)). Der zweite Ansatz bietet die Möglichkeit, eine beliebige Anzahl heterogener Rechnerarchitekturen in einem Netzwerk zu verflechten. Die einzelnen Reaktoren werden dabei zu selbstständigen Recheneinheiten, die verteilt im Netzwerk ausgeführt werden und Informationen austauschen. Gedacht ist hier in erster Linie an das Internet, über das sich (unpräzise formuliert) eine beliebige Zahl von Rechnern zusammenschließen lassen können. Gross et al. (2002) zeigen eine solche Architektur für verteiltes Rechnen im Internet. So lässt sich zum Beispiel ein automatischer Beweiser formen bestehend aus einem Cluster mit mehr als eine Millionen Rechnern.

Der Aufbau eines solchen Netzwerks kann eine aus der realen Chemie bekannten Rührkesselkaskade nachahmen (Abb. 3.4 (a)). Dabei sind alle Reaktoren linear verbunden. Der Abfluss eines Reaktors ist mit dem Zufluss des folgenden Reaktors verbunden. Der erste Reaktor der Kette hat wahlweise einen Zufluss (von Startklauseln) oder ist geschlossen. Die Klauseln im Abfluss des letzten Reaktors gehen verloren (Abb. 3.4 (b)). Denkbar wäre auch die Anordnung eines einzelnen Reaktors in einem Feld (Abb. 3.4 (c)). Der Fluss der Moleküle wäre in diesem Fall in mehr als eine Richtung orientiert.

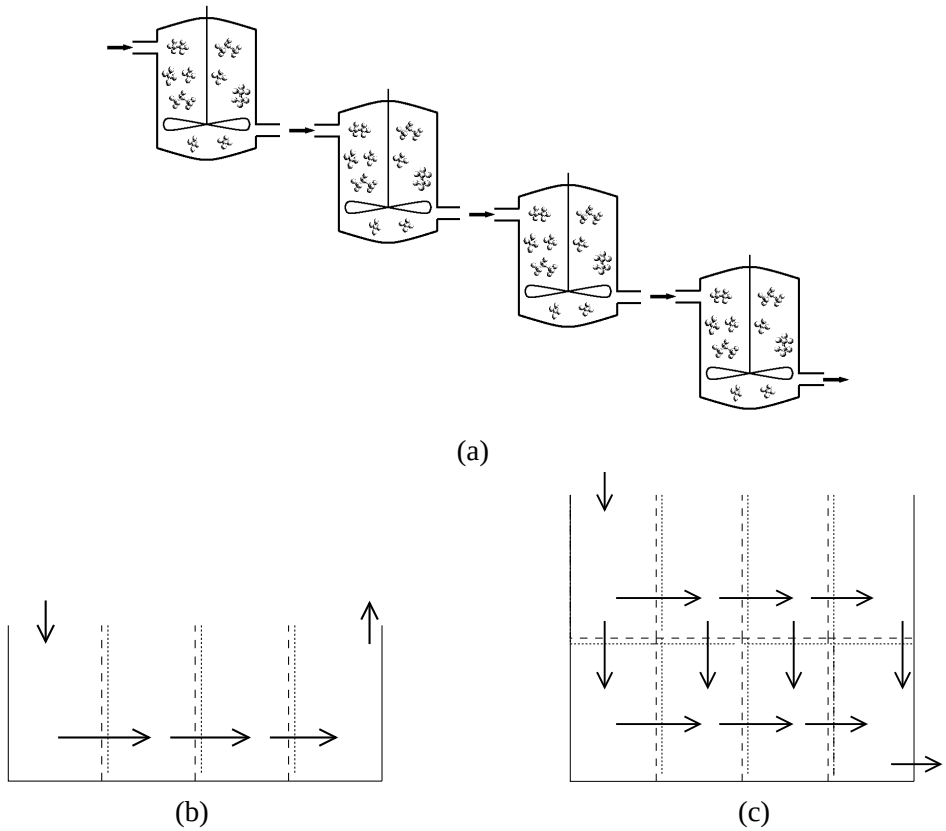


Abbildung 3.4: *Oben:* Rührkesselkaskade in einer realen Anordnung (vergleiche Westerterp et al. (1987)). *Unten links:* Eine lineare Anordnung von *RESAC*-Reaktoren. Semipermeable Membranen steuern den Molekülfluss. *Unten rechts:* Anordnung von Reaktoren in einem Feld.

3.3.1 Skizze des Verbindungsschemas und des Verbindungsaufbaus

Zu jedem Netzwerk gibt es einen ausgezeichneten Rechner, den *Administrator*. Er koordiniert den Verbindungsaufbau bzw. jede Verbindungsänderung. Auf diese Weise bestimmt er die Topologie des Netzwerks. Ein neuer Rechner meldet sich beim Administrator an, ein ausscheidender Rechner dort ab. Ausfallende Rechner werden ihm gemeldet, Nachrichten an alle im Netzwerk registrierten Rechner von ihm gesendet (broadcast message). An- und Abmeldung können zu jedem Zeitpunkt erfolgen, die Architektur erlaubt so die Bildung dynamischer Netzwerke. Während des Betriebs kann das Netzwerk dem Bedarf angepasst werden. Eine Dopplung des Administrators ist problemlos und sorgt für Ausfallssicherheit. Außerdem kann auf diese Weise der Administrator ausgewechselt werden, wodurch jederzeit neue Netztopologien programmiert werden können.

Vereinfachend dargestellt (siehe Abb. 3.5) wird auf Anfrage (1) eines ins vorhandene Netz zu integrierenden Rechners R_n mit der Registrierung, also der Speicherung seiner Zugangsports (2), und der Bekanntgabe dieser Ports in der Nachbarschaft, in die R_n integriert werden soll, geantwortet (3a). Zeitgleich bekommt auch R_n die für ihn relevanten Informationen über seine neuen Nachbarn mitgeteilt (3b). Im letzten Schritt nehmen die Nachbarrechner direkt Verbindung auf (4). Der Verbindungsabbau wird in Abbildung 3.6 verdeutlicht.

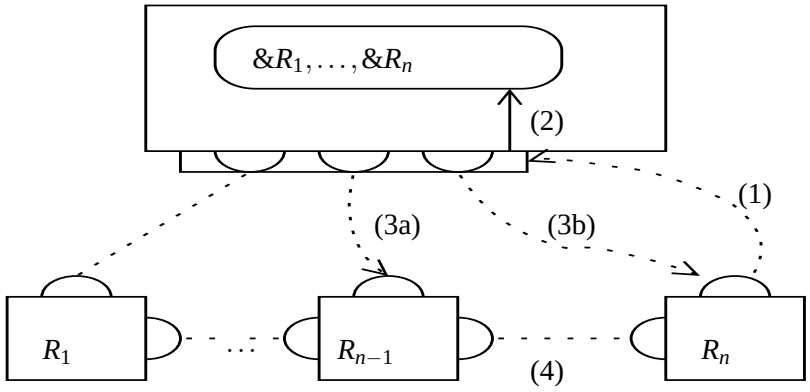


Abbildung 3.5: Grobes Schema des Verbindungsaufbaus bei linearer Reaktoranordnung. Strichlierte Pfeile zeigen gerichteten Informationsfluss entlang eines Kommunikationskanals an. Strichlierte Linien zeigen Kanäle, die im betrachteten Zeitfenster keinen Informationsfluss aufweisen. Drei Punkte deuten die Auslassung weiterer Reaktoren an. Die zeitliche Abfolge der Aktionen wird durch ihre Nummerierung festgelegt. Das Bild zeigt einen Administrator (oben) und $n-1$ Reaktoren im Netzwerk. Der anfangs isolierte Reaktor R_n wird durch folgende Aktionen zugeschaltet: **(1)** Verbindungsanfrage von R_n beim Administrator; **(2)** Speichern der Reaktorinformation von R_n im Administrator; **(3a)** Bekanntgabe des neuen Rechners bei R_{n-1}; **(3b)** Der Nachbarrechner R_{n-1} (Ende der Reaktorkette) wird R_n bekannt gemacht; **(4)** Verbindung R_{n-1}–R_n wird aufgebaut.

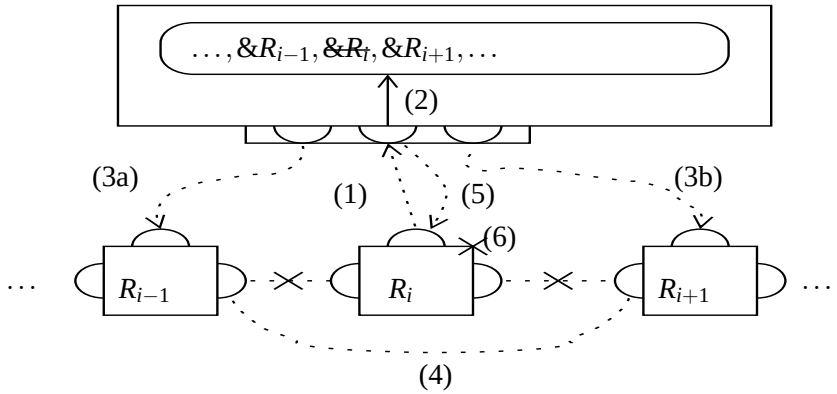


Abbildung 3.6: Grobes Schema des Verbindungsabbaus bei linearer Reaktoranordnung. Strichlierte Pfeile zeigen gerichteten Informationsfluss entlang eines Kommunikationskanals an. Strichlierte Linien zeigen Kommunikationskanäle, die im Rahmen des betrachteten Zeitfensters keinen Informationsfluss aufweisen. Drei Punkte deuten die Auslassung weiterer Reaktoren an. Die zeitliche Abfolge der Aktionen wird durch ihre Nummerierung festgelegt. Das Bild zeigt einen Administrator (oben), einen Reaktor R_i und seine benachbarten Reaktoren. R_i wird durch folgende Aktionen aus dem Netzwerk entfernt: **(1)** R_i macht dem Administrator seinen Wunsch zum Verbindungsabbau bekannt; **(2)** Aktualisierung der Reaktordatenbank des Administrators; **(3a)** und **(3b)** Bekanntgabe der veränderten Situation bei R_{i-1} und R_{i+1}; **(4)** Verbindungen zu R_i schließen, Verbindung R_{i-1}–R_{i+1} aufbauen; **(5)** Bestätigung des Verbindungsabbaus; **(6)** Nachbarverbindungen und Administrator-Verbindung schließen, Reaktor R_i herunterfahren.

Das gerade angedeutete Kommunikationsprotokoll setzt auf dem weit verbreiteten Verbindungs- bzw. Transportprotokoll TCP/IP auf. Damit ist die direkte Nutzung sowohl von LAN als auch des Internets möglich. Die Unterstützung dynamischer Netzwerke und die Toleranz gegenüber unvorhersehbaren Ereignissen wie Rechner- oder Verbindungsausfällen ermöglicht auch den effektiven Einsatz in stark reglementierten Batchsystemen wie dem LSF-Batchsystem (Zhou 1992), welches unter Anderem von dem europäischen Nuklearforschungszentrum CERN eingesetzt wird. Es sei darauf hingewiesen, dass alle verwendeten Netzwerkprotokolle zusammen mit der realisierten Portverwaltung keine Anforderungen an den Standort der Partner-Rechner stellen und darüber hinaus mehrere Prozesse auf ein und demselben Rechner unterstützen. So ist die Parallelisierung von Experimenten auf nur einem physikalischen Rechner, der dabei kein Parallelrechner sein muss, genauso möglich wie das Einbinden eines sich beispielsweise in Korea befindlichen Rechners.

3.3.2 Datenaustausch

Der Austausch von Daten – hier Molekülen – zwischen Rechnern eines Netzwerks soll für ein laufendes Experiment unsichtbar erfolgen. Für einen Reaktor ist es unerheblich, ob er isoliert, d. h. nicht parallel zu anderen, oder im Verbund mit anderen betrieben wird – solange Moleküle zu- und abfließen können. Genauso ist die gegebene Netztopologie, falls es Parallelverarbeitung gibt, uninteressant und daher zu verbergen. Objektorientiertes Programmdesign gibt hierzu die Möglichkeit in Form der Spezifikation von Interfaces[3]. Mit diesem Werkzeug kann die Anbindung eines Netzwerks unabhängig vom eigentlichen Reaktor realisiert werden; die nächsten Abschnitte demonstrieren dies. Diese softwareseitige Abkapselung ermöglicht eine flexible, austauschbare Implementierung. Die Zuteilung der Partner-Rechner im Netzwerk erfolgt wie oben beschrieben durch den Administrator, der somit die Netzwerktopologie festlegt.

Eine Instanz der Klasse `molecule experiment` tauscht die den Reaktor verlassenden und die in diesen eintretenden Moleküle über die Schnittstellen `export` und `import` des (abstrakten) Interfaces `reactorIO` aus. Beim isolierten Einzelreaktor oben in Abb. 3.7 werden die Anfragen durch die Klasse `reactorStandalone` beantwortet, und zwar auf schon beschriebene Art und Weise (siehe insbesonders Algorithmen 3.1-3.3). Im Parallelbetrieb (unten im Bild) hingegen wird das Interface durch eine Klasse implementiert, die auf die Anfragen durch Interaktion mit parallel gestarteten `experiment`-Instanzen reagiert. Wie die Moleküle dabei ausgetauscht werden, ist für diese Klasse bedeutungslos, sie greift auf die Schnittstellen des Interfaces `netConnector` zurück, das sowohl für die Netzanbindung als auch für die Paketierung der Daten verantwortlich ist.

Abbildung 3.7 zeigt die Implementierung des Interfaces `netConnector` durch eine gepufferte Datenhaltung (`bufferedTransmitterReceiver`). Für die abfließenden Moleküle übernimmt die Klasse `outputReservoir` die Akkumulation. Ist der Puffer voll, löst sie durch die Schnittstelle `transmit` die Übertragung des Datenvektors aus. Umgekehrt übernimmt `inputReservoir` die Speicherung eines ankommenden Molekülvektors. Ein vorgeschalteter Konverter wandelt je nach Übertragungsrichtung einen Molekülvektor in ein für die Übertragung geeignetes Paket (engl. package) oder umgekehrt.

[3] Im Sprachgebrauch des objektorientierten Designs bezeichnet ein Interface eine Spezifikation von Schnittstellen, die von einer das Interface implementierenden Klasse angeboten werden müssen. Die konkrete Realisierung der Anforderungsbeschreibung ist dabei unerheblich.

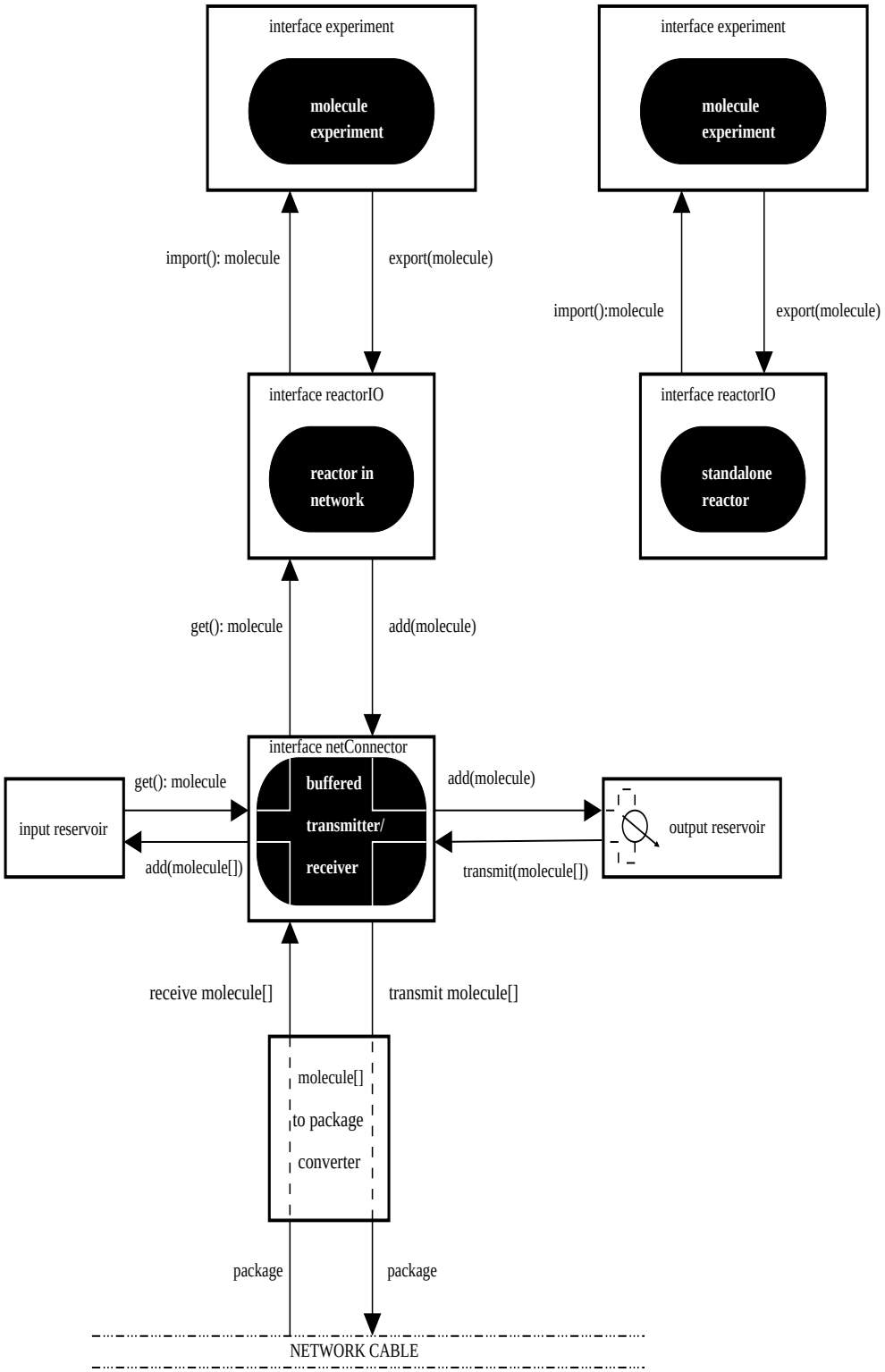


Abbildung 3.7: Schema der Datenflüsse in *RESAC*: Für die Klasse `molecule experiment` ist es unerheblich, ob das Interface `reactorIO` Verbindungen zu weiteren Reaktoren aufbaut oder ob es isoliert, d. h. nicht parallel zu anderen abläuft. Weitere Erklärungen siehe Text.

3.3.3 Demonstration zweier paralleler Experimente

Der parallele Betrieb mehrerer miteinander interagierender Reaktoren soll abschließend an zwei Beispielen demonstriert werden. Dabei handelt es sich beim ersten Beispiel um den einfachen Fall, dass keine Reaktionen stattfinden. Hier wird die Wirkung der Zu- und Abflusskopplung unverfälscht deutlich. Das zweite Beispiel zeigt eine etwas komplexere Dynamik. Eine Anwendung beim automatischen Beweisen findet sich in Kapitel 6 dieser Arbeit.

Beispiel 3.3.1
Gegeben sind die folgenden vier Molekülsorten:

$$M_1 = \mathtt{P(a)} \qquad M_2 = \mathtt{Q(a)} \qquad M_3 = \mathtt{R(a)} \qquad M_4 = \mathtt{S(a)}$$

Es gibt in *RESAC* zu $M_1, \ldots, M_4$ keine Molekülpaarung, infolge derer eine Reaktion abläuft. Vier Reaktoren $T_1, \ldots, T_4$ sind bei diesem Experiment in der oben beschriebenen Weise linear verkoppelt (siehe Abb. 3.4 (b)). Anfangs befindet sich in jedem der Reaktoren genau eine Molekülsorte, nämlich Molekül M_i in Reaktor T_i für $i = 1, \ldots, 4$.

Die Dynamik der parallelen Interaktion ist in Abbildung 3.8 aufgezeigt. Die einzelnen Molekülsorten „durchlaufen“ alle nachgeschalteten Reaktoren, es findet sich M_4 ausschließlich in T_4, bevor es verdrängt wird, M_3 in T_3 und T_4. Molekül M_2 taucht in den Reaktoren T_2 bis T_4 auf. Sorte M_1 durchströmt alle Reaktoren und ersetzt dabei im Laufe der Zeit alle anderen Sorten. Die in der Abbildung gezeigten Kurvenverläufe entsprechen der Lösung des zugehörigen Differenzialgleichungssystems. Sei im Folgenden $1 \leq i, j \leq n$. Bezeichnet $x_{i,j}(t)$ die Konzentration von M_i in Reaktor j zum Zeitpunkt $t \geq 0$, so gilt für die Verallgemeinerung dieses Experiments auf n Reaktoren und Zuflussrate Ψ, die für alle Reaktoren vereinbart wird:

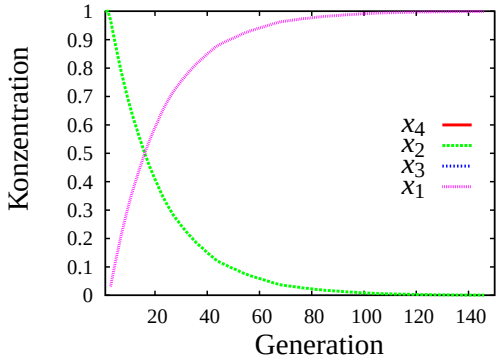
$$\frac{d}{dt}x_{i,j} = \begin{cases} 0 & \text{für } i > j, \\ 0 & \text{für } i = j = 1, \\ -x_{i,i}(t)\,\Psi & \text{für } i = j \neq 1, \quad (*) \\ x_{i,j-1}(t)\,\Psi \;-\; x_{i,j}(t)\,\Psi & \text{für } i = 1, \ldots, j-1. \quad (**) \end{cases} \tag{3.5}$$

Dabei beschreibt (*) die Veränderung von M_i im Reaktor T_i, dem „Geburtsreaktor“. Jeglicher Zufluss kann sich hier nur negativ auf x_i auswirken, denn er enthält keinesfalls M_i-Moleküle. Die Verdrängung geschieht proportional zu x_i. In nachgeschalteten Reaktoren gibt es dagegen M_i-Zufluss: Der Anteil des Zuflusses, der M_i-Moleküle beschreibt, kommt x_i im Folgereaktor zugute (**). Allerdings verdrängt er in diesem Folgereaktor auch M_i-Moleküle, und zwar proportional zu ihrer dortigen Konzentration.

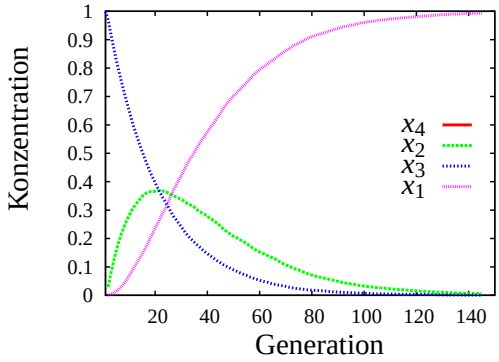
Die Anfangsbedingungen sind spezifiziert durch

$$x_{i,j}(0) = \begin{cases} 1 & \text{falls } i = j, \\ 0 & \text{sonst.} \end{cases} \tag{3.6}$$

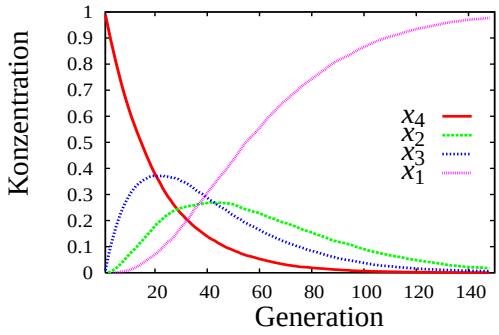
Reaktor T_2:



Reaktor T_3:



Reaktor T_4:



Analytische Lösung:

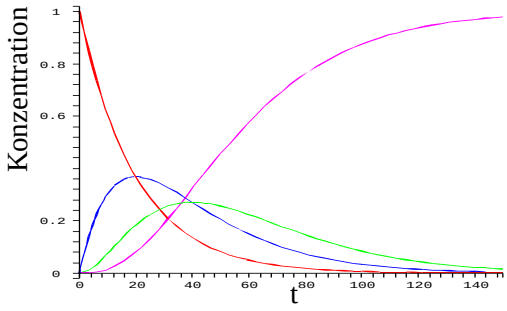


Abbildung 3.8: Paralleles Experiment in *RESAC* nach Beispiel 3.3.1. Verbunden sind die 4 offenen Reaktoren $T_1,\ldots,T_4$ unter Eduktersetzung und 5%-igem Zufluss. Gezeigt sind die Konzentrationsverläufe in den Reaktoren $T_2 - T_4$. Im ersten Reaktor gibt es zu jedem Zeitpunkt nur Moleküle der Sorte M_1, es fließen auch nur solche zu und ab. In T_2 ersetzt dieser Zufluss die vorhandenen M_2-Moleküle. Anfangs fließen M_2-Moleküle ab, im zunehmenden Maße dann auch M_1-Moleküle. Schließlich erreichen den dritten Reaktor nur noch M_1-Moleküle. Reaktor T_3 gibt analog die Moleküle M_3, M_2 und M_1 an den letzten Reaktor weiter. Die Gewichtung ist wiederum abhängig von der Zeit. Schließlich haben sich die fortwährend in den ersten Reaktor einströmenden Moleküle der Sorte M_1 in allen Reaktoren komplett durchgesetzt. Die unteren Graphen zeigen die Sortenverteilung im letzten Reaktor – links durch *RESAC* experimentell bestimmt, rechts analytisch durch Lösung des Differenzialgleichungssystems ermittelt.

Für $n = 4$ und $\Psi = 0.05$ lässt sich dieses Differenzialgleichungssystem analytisch lösen. Es ergeben sich für die Molekülverteilung im letzten Reaktor:

$$\begin{aligned} x_{1,4}(t) &= 1 + \frac{1}{800} e^{\left(-\frac{1}{20}t\right)}(-t^2 - 40t - 800) \\ x_{2,4}(t) &= \frac{1}{800} e^{\left(-\frac{1}{20}t\right)} t^2 \\ x_{3,4}(t) &= \frac{1}{20} e^{\left(-\frac{1}{20}t\right)} t \\ x_{4,4}(t) &= e^{\left(-\frac{1}{20}t\right)} \end{aligned}$$

Beispiel 3.3.2
In einem Experiment sei Eduktersetzung vereinbart und die beiden folgenden Molekülsorten gegeben:

$$M = \mathtt{P(a)Q(a)} \qquad N = \overline{\mathtt{P(X)}}.$$

Diese Sorten werden zu Anfang des Experiments auf sechs Reaktoren $T_{Start}, T_1, \ldots, T_5$ verteilt. Und zwar in dieser Weise: Moleküle der Sorte M befinden sich ausschließlich im ersten Reaktor, der im Weiteren *Starter* genannt wird. Keine andere Molekülsorte ist in diesem vertreten. Der Inhalt der übrigen Reaktoren sind Moleküle der Sorte N. Bereits diese einfache Anordnung entwickelt eine erstaunliche Dynamik, wenn die Reaktoren parallel arbeiten und in der oben beschriebenen linearen Weise verkoppelt sind. Die Reaktoren seien also in Reihe geschaltet, wobei der Starter T_{Start} das Anfangsglied dieser Kette bildet.

Das durch Resolution induzierte „Netzwerk“ zu den Sorten M und N ist festgelegt durch die einzig mögliche Reaktion

$$M \text{ x } N \rightarrow L, \text{ wobei } L = \mathtt{Q(A)}. \tag{3.7}$$

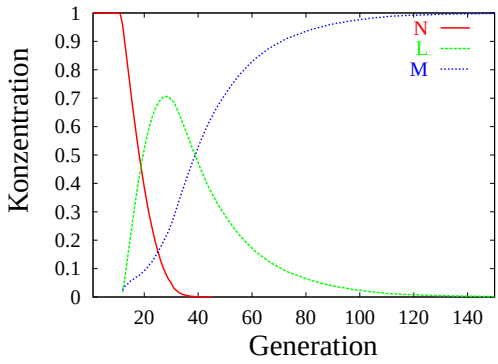
Eine Anwendung der Resolution zwischen anderen Molekülsorten ist – genauso wie autokatalytische Kopplungen – nicht möglich. Im ersten Reaktor T_{Start} erfolgt demnach keine einzige produktive Kollision. Gleiches gilt für die restlichen Reaktoren $T_1, \ldots, T_5$. Isoliert betrieben findet demnach in keiner der Reaktoren auch nur eine Reaktion statt.

Wird jedoch die oben beschriebene Kaskadierung zugrunde gelegt, so fließen M-Moleküle in den ersten Reaktor T_{Start}. Der dadurch verursachte Abfluss, zusammengesetzt ausschließlich aus M-Molekülen, gelangt in den folgenden Reaktor T_1, dessen Dynamik in den folgenden Abschnitten analysiert wird. Durch das Zusammentreffen von M- und N-Molekülen in diesem Reaktor kann nun die in Gleichung 3.7 beschriebene Reaktion stattfinden. Molekül L wird produziert. Und zwar umso mehr, je mehr Moleküle der Sorte M eingeflossen sind. Kollisionen mit L-Molekülen verlaufen elastisch.

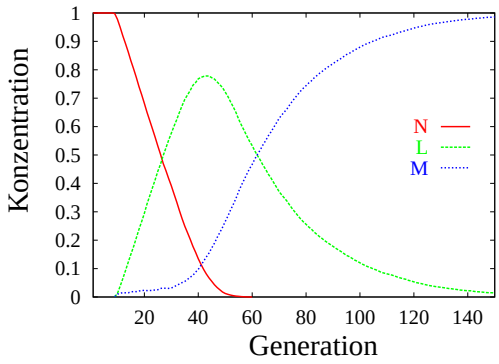
Abbildung 3.9 zeigt den Eintritt eines verstärkenden Effektes: Der anfängliche Anstieg der L-Moleküle in T_1 erfolgt stärker als es der gleichmäßige Zufluss (von in diesem Fall 5%) vermuten ließe. Der Grund liegt in der Eduktersetzung. In rund der Hälfte aller Reaktionen zwischen Molekülen der Sorten M und N bleibt ein M-Molekül bestehen, welches in weiteren Reaktionen wieder durch produktive Reaktionen zur erneuten Bildung von L führt. In dieser ersten Phase werden viele der einströmenden M-Moleküle in Reaktionen verbraucht (es gibt ein Überangebot an N-Molekülen). Ähnliches gilt für deren Reaktionspartner, den Moleküle der Sorte N, sie werden in gleichem Maße „verstoffwechselt“.

Phase 2 beginnt in etwa, wenn der Anstieg der L-Moleküle und der Rückgang der N-Moleküle dazu geführt haben, dass beide Sorten jeweils eine Konzentration von knapp 50% aufweisen. Es gibt nun kein Überangebot an N-Molekülen mehr. Es wird zunehmend schwerer für die einfließenden M-Moleküle, Reaktionspartner zu finden. Sie beginnen sich deutlich abzusetzen. Da auch die L-Moleküle durch den Zufluss regelmäßig ersetzt werden, ist ihre maximale Konzentration erreicht, noch bevor das letzte N-Molekül aus dem Reaktor verdrängt wurde. Ihre Konzentration sinkt im weiteren Verlauf.

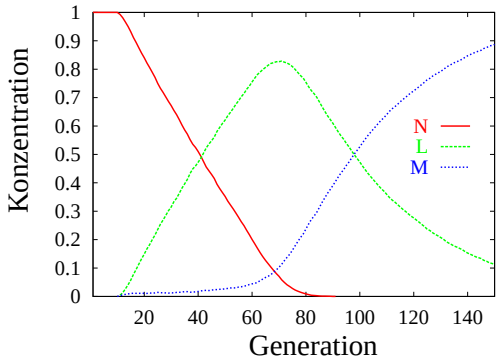
Reaktor T_1:



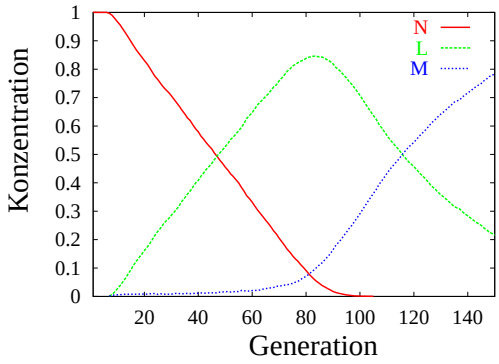
Reaktor T_2:



Reaktor T_3:



Reaktor T_4:



Reaktor T_5:

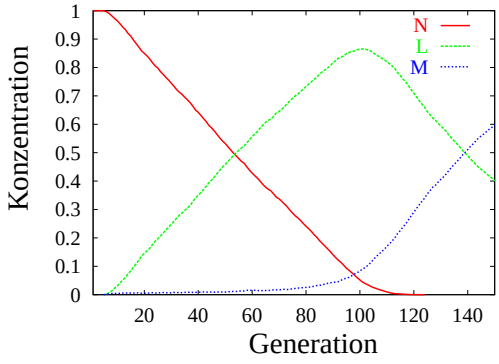


Abbildung 3.9: Paralleles Experiment in *RESAC* nach Beispiel 3.3.2. Verbunden sind die sechs Reaktoren $T_{Start}, \ldots, T_5$ unter Eduktersetzung. Reaktor T_{Start} wurde in Generation 11 geöffnet bei 5%-igem Zufluss. Das Verhalten lässt sich in drei Phasen unterteilen. Erklärung siehe Text.

Die letzte Phase ist gekennzeichnet durch die Abwesenheit von N-Molekülen. Diese sind in den beiden vorhergehenden Phasen entweder „verstoffwechselt“ oder durch Zufluss verdrängt worden. Es kann nun wie zu Anfang des Experiments keine Reaktion mehr stattfinden. Durch den gleichmäßigen Zufluss von M-Molekülen aus T_{Start} werden alle Moleküle schrittweise ersetzt, so dass am Ende nur noch Moleküle des Typs M im Reaktor T_1 vorhanden sind.

Diese drei Phasen finden sich qualitativ in jedem der T_{Start} nachgeschalteten Reaktoren wieder wie Abb. 3.9 zeigt. Je weiter aber ein Reaktor vom Starter T_{Start} in der Kaskade entfernt liegt, desto gedehnter sind seine Phasen (unter Verschiebung der folgenden Phasen). Am Beispiel der zweiten Phase ist das in Abb. 3.10 besonders deutlich zu sehen. Ursache ist der inhomogene Zufluss der Reaktoren $T_2, \ldots, T_5$, der nicht nur aus M-Molekülen besteht, sondern jeweils das Konzentrationsgemisch des Vorgängerreaktors widerspiegelt. Es fließen $T_2, \ldots, T_5$ auch L-Moleküle zu; die erreichbare Maximalkonzentration von L-Molekülen wird so von Reaktor zu Reaktor gesteigert.

Bemerkung: Die Wahl der Eduktersetzung sorgt für gleichmäßigen Zufluss in allen Reaktoren, da das Abflussvolumen nicht reaktionsabhängig ist (wie bei der freien Ersetzung), sondern allein durch den Zufluss geregelt wird.

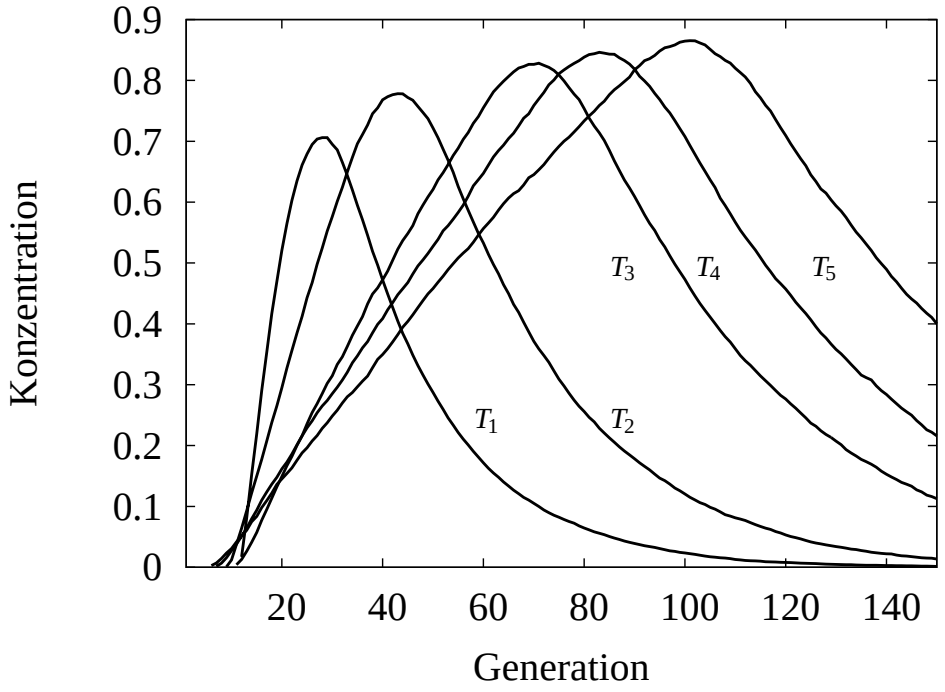


Abbildung 3.10: Die Konzentrationsverläufe der L-Moleküle aller in Reihe geschalteten Reaktoren zum Beispiel 3.3.2 im direkten Vergleich. Deutlich wird die zunehmende Dehnung der 2. Phase.

3.4 *RESAC* im Kontext anderer Arbeiten

RESAC grenzt sich einerseits ab von den analogen Künstlichen Chemien, die ein zumindest in Teilbereichen genaues Modell realer Chemien entwerfen wollen. Mayer und Rasmussen (1998), Bersini (2000) sowie Benkö et al. (2003) zeigen Ansätze in dieser Richtung. Betrachtet man andererseits die Künstlichen Chemien, die wie *RESAC* eher abstrakt angelegt sind, so unterscheidet sich *RESAC* von denjenigen, die Modelle entwerfen und analysieren zur Untersuchung von zahlreichen Aspekten der Evolution: Selbstorganisation (Banzhaf 1993; Kauffman 1993; McCaskill et al. 1994; Adami 1995; Fontana und Buss 1996), Symbiose und Parasitentum (Boerlisr und Hogeweg 1991; Ray 1991), Autopoiesis (Varela et al. 1974; Luisi 1991; McMullin und Varela 1997), Replikation (Pargellis 1996; Ono und Ikegami 1999; Hutton 2002; Hutton 2003), Mutation (Ikegami und Hashimoto 1995; Lenski et al. 1999), Diversifikation (Entstehung des Artenreichtums) und Nischenbildung (Dittrich et al. 2000), gekoppelter sowie ungekoppelter Wettbewerb, Rückkopplungen, Zyklen und Informationsintegration (Eigen und Schuster 1977; Farmer et al. 1986; Hofbauer und Sigmund 1988; Kauffman 1993; Banzhaf 2002). Evolutionsexperimente in *RESAC* werden in Kapitel 5 angesprochen.

Andere abstrakte Künstliche Chemien dienen der allgemeinen Modellbildung: Bekannt sind u. a. Modelle soziologischer (Szuba 1997), astrobiologischer (Centler et al. 2003) oder technischer Systeme (nebenläufige Prozesse [Berry und Boudol (1992)]). Des Weiteren gibt es eine Vielzahl von Wachstumsmodellen in der Biologie (Zellwachstum und -teilung [Furusawa und Kaneko (1998), Ono und Ikegami (2000), Madina et al. (2003)], Neuronen- und Synapsenwachstum [Astor und Adami (2000)]).

RESAC als resolutionsbasierte Künstliche Chemie tendiert zur angewandten Problemlösung. Neben dem automatischen Beweisen und dem Lösen von Entscheidungsproblemen oder Konstruktionsaufgaben gehört die Optimierung zum abgedeckten Anwendungsbereich. Nach (Dittrich 2001) gibt es prinzipiell zwei Möglichkeiten, Optimierungsprobleme im Rahmen einer Künstlichen Chemie zu bearbeiten. Einerseits repräsentiert jedes Molekül des Reaktors eine potenzielle Lösung. Reaktionen führen zu Veränderungen einzelner Lösungskandidaten und vervielfältigen gute Lösungen. Solche Chemien ähneln oftmals den Evolutionären Algorithmen (Bäck et al. 1997). Durch die Einführung „aktiver“ Moleküle, die quasi wie Maschinen (Operatoren) andere Moleküle bearbeiten und so Vorgänge wie Selektion, Mutation und Rekombination bewerkstelligen, wird der Optimierungsvorgang zwar chemischer, doch bleiben die verwendeten Werkzeuge die gleichen.

Beim zweiten Ansatz wird der komplette Reaktor als mögliche Lösung aufgefasst. Diese Moleküle können beispielsweise problemspezifisch verbunden sein. Eine Reaktion hat dann Einfluss auf die Struktur der Moleküle und/oder deren Verbindungen. Ein bekanntes Beispiel dafür ist das Modell von (Kanada und Hirokawa 1994), das der übernächste Absatz diskutiert.

RESAC benutzt hier einen neuen Ansatz: Zwar beschreibt jedes Molekül der Population einen Kandidaten, aus dem eine Lösung heranwachsen kann, der aber im Gegensatz zum ersten Ansatz noch keine strukturell gültige Lösung darstellt, also nicht einmal eine sehr schlechte. Durch Interaktion der Moleküle aggregiert sich Information, Lösungen ergeben sich aus dem emergenten Verhalten dieses dynamischen Prozesses. Moleküle sind auch nicht immer eindeutig charakterisierbar als Operatoren oder Operanden. Die zwei Moleküle einer Reaktion tragen beide Information, die zusammengeführt wird. Dies geschieht in einem Prozess, der wesentlich komplexer ist, als es eine

Rekombination üblicherweise ist. In *RESAC* ändert sich nicht nur der Status eines Moleküls oder seine Verbindungen (vgl. (Hutton 2003)), sondern die dem System zugrunde liegende Resolutionsanwendung modifiziert die gesamte Struktur auf *nicht-deterministische* Art und Weise. Dabei bleibt die Änderung nicht lokal beschränkt wie bei den Rekombinationsoperatoren Evolutionärer Algorithmen, sondern erstreckt sich durch den Unifikationsprozess auf das ganze Molekül. Eine Mutation im herkömmlichen Sinne bzw. eine Selektion findet nicht statt.

Insbesondere die Arbeiten von Yasusi Kanada, im Wesentlichen (Kanada und Hirokawa 1994; Kanada 1995b; Kanada 1995a), sind im Bereich Optimierung und Problemlösung im Rahmen von Künstlichen Chemien zu nennen. Die von ihm entwickelten CCM-Systeme (CCM steht für engl. chemical casting model) nutzen zur Lösung kombinatorischer Optimierungs- oder Konstruktionsprobleme emergentes und selbstorganisatorisches Verhalten aus. Der Reaktorinhalt, nämlich Atome und Verbindungen zwischen diesen, beschreibt als Ganzes eine korrekte Lösung des Problems. Regeln ähnlich den Produktionsregeln in Expertensystemen, jedoch einfacher strukturiert, steuern den Umbau des Reaktors. Jedes „Teilgemisch“, das mit dem Muster des Bedingungsteils einer Regel übereinstimmt, wird entsprechend der Regel ersetzt. Zu beachten ist die etwas andere Sichtweise im Vergleich zur Künstlichen Chemie *RESAC*, in der sich Moleküle bewegen und bei Kollision eine Reaktion auslösen können. Bei Kanada sind die Regeln dagegen nicht Teil der Population, es wird auf nicht-deterministische Weise versucht, sie mit den Atomen zur Deckung zu bringen.

Ob eine erfolgreiche Überdeckung (engl. matching) eine Regelanwendung tatsächlich nach sich zieht, hängt in CCM noch von einer anderen Bedingung ab: Jeweils vor und nach einer hypothetischen Konfigurationsänderung wird ein Energiemaß berechnet. Nur wenn eine Regelanwendung zu einer energetisch günstigeren Konfiguration führt, wird sie tatsächlich durchgeführt. Dadurch wird impliziert ein Fitnessmaß eingeführt, das es ermöglicht, den Problemlösungsprozess zu steuern. Die Verwendung eines solchen lokal errechneten Entscheidungskriteriums führt zur Selbstorganisation des Systems.

Einerseits lässt sich auf diese Weise die Effizienz bei Optimierungsproblemen steigern (gerichtete Suche) und zudem das Gesetz der Thermodynamik mit ins Kalkül ziehen.[4] Andererseits aber besteht die Gefahr, dass die Suche in lokalen Maxima beendet wird.[5] In der Tat stößt Kanada (1995b) auf dieses Problem und führt zusätzliche Mechanismen[6] ein. Der Begriff der Emergenz kann sich in solchen Fällen nicht mehr allein auf die Vielzahl einzelner 1:1 Interaktionen abstützen.

Das Entscheidungskriterium ist darüber hinaus meist problemspezifisch. Damit ist das Verfahren nicht ohne Modifizierung des Systems allgemein anwendbar. Ein weiterer Nachteil besteht darin, dass ein solches Energiemaß vom Anwender – dem Ersteller des Regelsatzes – beliebig ausgestaltet werden kann. Eine Umsetzung in reale Modelle des molekularen Rechnens wäre – falls überhaupt möglich – äußerst komplex und zudem erneut problemspezifisch.[7] Kanada spricht diesen Aspekt nicht an, eine Umsetzung ist nicht Ziel seines Ansatzes.

[4] Hier wird die Zunahme der Entropie symbolisiert.

[5] Die Wahrscheinlichkeit dafür ist abhängig von der Struktur des vorliegenden Problems (Kanada und Hirokawa 1994). In dem Maße wie lokale Fitnessverbesserungen auch globale Fitnessverschlechterungen bewirken können, wird die alleinige Optimierung nach der Strategie „Bergsteigen“ (engl. hill climbing) verhindert.

[6] Schrittweitenerhöhung, Simulated Annealing

[7] Ein Weg, ein Energiemaß zu implementieren, wäre verschiedene Konformationen von Molekülen zu betrachten. Eine Regelanwendung würde in diesem Fall dann durchgeführt, wenn sich das entstehende Molekül als energetisch

Die Umsetzung resolutionsbasierter Künstlicher Chemien durch molekulares Rechnen ist ein verlockendes Ziel. Die enorme Zahl von Molekülen, die in kurzer Zeit parallel reagieren und die hohe Speicherdichte beispielsweise von DNA-Molekülen sind der Grund. Die chemische Metapher der KC erleichtert die Überführung in eine Parallelisierung auf einem natürlichen System. Dabei übertragen sich die Prinzipien von Selbstorganisation, Wettbwerb und Emergenz auf das massiv parallel arbeitende System. (Banzhaf 1992) beleuchtet diesen Aspekt. Der Ausblick der vorliegenden Dissertation geht näher auf die Übertragung der KC ein. Um diese Übertragung nicht zu gefährden, verzichtet *RESAC* auf ein irgendwie geartetes Energie- oder Informationsmaß zur Bestimmung der Fitness. Ein weiterer Grund ist die Festlegung auf eine unveränderliche, problemunabhängige Regelmenge R, die die Konstruktion Künstlicher Chemien – wie anfangs des Kapitels dargestellt – erleichtert. Schlussendlich wird der Verbleib in lokalen Maxima verhindert.

Davon abgesehen muss man sich – falls allgemeine Problemlösung angestrebt ist – fragen, ob überhaupt ein problemunabhängiges Maß für die Güte von Molekülen oder Reaktionen definiert werden kann. Allein schon für das automatische Beweisen (siehe Kapitel 6) ist kein allgemein anerkanntes Kriterium bekannt. Das von Nordin und Banzhaf (1997) verwendete Längenkriterium (in der Auswirkung ähnlich der Unit-Resolution, vgl. Unterkapitel 1.3.3) etwa ist nicht widerlegungsvollständig und daher im Allgemeinen nicht in Theorembeweisern einsetzbar. Tabelle 6.1 (Seite 114) zeigt einen Beweis, in dem die Klauseln erst einmal länger werden. bevor sie sich wieder verkürzen.

3.5 Zusammenfassung

RESAC beschreibt eine in Form des Tupels (A,S,R) gegebene Künstliche Chemie. Es folgt eine Aufzählung wichtiger Eigenschaften der drei Komponenten, anhand derer sich das vorliegende System von anderen unterscheidet:

Moleküle aus S:
- Alle Moleküle haben fest vorgegebene Struktur. Sie ist festgelegt durch die Syntax der Prädikatenlogik erster Ordnung.
- Es gibt keine ausgezeichneten Moleküle, keine ausgewiesenen Operatoren oder Operanden.
- Es ist kein Gütemaß (Fitness) auf den Strukturen definiert.

Reaktionsregelmenge R:
- Genau eine problemunspezifische Reaktionsregel bestimmt den Ausgang einer Kollision zwischen zwei Molekülen: die Resolutionsregel (möglicherweise erweitert durch Faktorisierung, siehe Unterkapitel 6.3).
- Die Reaktionsberechnung ist ein nicht-deterministischer Prozess.
- Die Reaktionsberechnung ist ein komplexer Prozess, nicht mit einfacher Mutation oder herkömmlicher Rekombination vergleichbar.

günstiger erweist. Die Kreation realer Moleküle mit problemspezifischen Konformationsregeln erscheint (zur Zeit) unvorstellbar.

- Es gibt keine Güte-Selektion, in Folge derer kollidierende Moleküle reagieren oder nicht. Ob eine Kollision produktiv ist, hängt allein von den Strukturen der beteiligten Moleküle ab. Reaktionsprodukte werden immer in die Population übernommen, dies geschieht unabhängig von ihrer Güte.

Algorithmus *A*:

- Jeder Punkt des Suchraums (Molekül) beinhaltet eine vollständige potentielle Lösungsinstanz, ein Hilfskonstrukt oder ein Faktum. Potentielle Lösungsinstanzen repräsentieren aber im Allgemeinen keine Lösung, sondern stellen vielmehr eine Hypothese dar, die es im dynamischen Prozess zu Erhärten gilt.
- Es werden gut gerührte Reaktoren konstanter Größe modelliert, so genannte Rührkessel (CSTR).
- Die Reaktoren haben entweder eine offene oder geschlossene Architektur.
- Es sind zwei Ersetzungsvarianten vorgesehen. Eine Diskussion findet sich in Kapitel 4.

Parallelisierung: Eine Parallelisierung des Systems wurde durchgeführt auf der Ebene von Subreaktoren und implementiert.

Anwendungsbereich: Alle im Prädikatenkalkül darstellbaren Probleme gehören zum Anwendungsbereich von *RESAC*.

In Kapitel 5 wird auf Evolutionsprobleme eingegangen, in Kapitel 6 das automatische Beweisen thematisiert. Kapitel 7 und 8 zeigen, wie mit Hilfe des hier spezifizierten Systems Künstliche Chemien programmiert werden können. Entscheidungs- und Optimierungsprobleme sind dann Gegenstand der Betrachtung. Dabei lösen sich die dargestellten Konzepte von der Semantik der Prädikatenlogik.

Kapitel 4

Diskussion der Ersetzungsmethodik

Bei der Definition einer Reaktionsgleichung innerhalb einer konstruktiven Künstlichen Chemie stellt sich grundsätzlich die Frage, wie mit einem neu produzierten Molekül verfahren wird. Durch *RESAC* werden nur Reaktoren mit unveränderlicher Größe und konstanter Dichte konstruiert. Eine Beibehaltung aller schon bestehenden sowie der neu gebildeten Moleküle ist daher nicht möglich. Zu bemerken ist, dass in dieser abstrakten KC die Reaktionsprodukte nicht durch Umordnung der elementaren Bestandteile der Edukte entstehen. Bei einer produktiven Reaktion gibt es daher für den Fall, dass die angewandte Reaktionsregel mehr Produkte als Edukte aufweist, einen tatsächlichen Zugewinn. In *RESAC* ist dies der in Gleichung 3.1 definierte *Reaktionszugewinn* R_Δ, eine Resolvente der beiden Edukte. Dieser Reaktionszugewinn kann entweder – in irgendeiner Form – Edukte ersetzen oder aber ein beliebiges anderes Molekül des Reaktionsgemisches verdrängen. Anders formuliert geht es um die prinzipielle Festlegung, welche der Edukte autokatalytisch wirken. Dieser Fragenkomplex wird hier unter der *Wahl der Ersetzungsmethode* verstanden.

In Kapitel 3.1.2 wurden zwei verschiedene Ersetzungsvarianten eingeführt, die zu folgenden alternativen Reaktionsregeln führen (in chemischer Notation):

Eduktersetzung: $R_{\text{Edukt}} : S_1 + S_2 \longrightarrow S_i + S_3,\ i \in \{1,2\}.$

Freiersetzung: $R_{\text{Free}} : S_1 + S_2 \longrightarrow S_1 + S_2 + S_3,$

S_3 entsteht durch Reaktion von S_1 und S_2 und bezeichnet den Reaktionszugewinn. Die Reaktionsvorschriften sind so zu verstehen, dass es bei der Produktion von S_3 in Folge einer produktiven Kollision unerheblich ist, in welcher Reihenfolge die Edukte zusammentreffen. S_3 ist das einzige Molekül, dass keine katalytische Wirkung entfaltet.

Es drängt sich die Frage auf, ob und wie sich die beiden Ersetzungsvarianten bezüglich ihrer Auswirkung auf die Konzentrationsverläufe der im Reaktor befindlichen Moleküle unterscheiden. Diese Frage ist bisher in der Literatur nicht nachhaltig thematisiert worden; am Ende dieses Kapitels folgt diesbezüglich eine Einordnung. Offensichtlich werden durch die beiden Ersetzungsvarianten unterschiedliche Wettbewerbssituationen geschaffen. Eine Analyse der Ersetzungsmethodik soll nun in dieser Dissertation erfolgen, da es sich hierbei um einen zentralen Parameter des Systems *RESAC* handelt. Dazu wird eine einfach strukturierte Problemstellung identifiziert und analytisch untersucht.

4.1 Das Basisexperiment für geschlossene Reaktoren

Bei den beiden oben angegebenen Reaktionsvorschriften handelt es sich um allgemeine Reaktionsregeln, also gewissermaßen um Schablonen, denn für S_1 und S_2 können alle Moleküle der KC eingesetzt werden. Abstrahiert man an dieser Stelle davon, dass sich S_3 in *RESAC* als Funktion von S_1 und S_2 ergibt, und instanziiert S_1, S_2 und S_3 mit den spezifischen Molekülsorten s_1, s_2 und s_3, so gelangt man zu folgender *expliziten* Reaktion über einer dreielementigen Molekülmenge S_B:

$$S_B = \{s_1, s_2, s_3\}$$

$$s_1 + s_2 \longrightarrow (s_3) + s_i,\ i \in \{1,2\} \quad \text{(Eduktersetzung)} \tag{4.1}$$

$$s_1 + s_2 \longrightarrow (s_3) + s_1 + s_2 \quad \text{(Freiersetzung)} \tag{4.2}$$

Die Klammerung kennzeichnet den Reaktionszugewinn. Notiert man nur den Zugewinn, so kann man (4.1) und (4.2) vereinfachend zusammenfassen. Benennt man schließlich s_3 der Übersichtlichkeit halber um in s_o (o steht für engl. output und zeichnet diesen Molekültyp gegenüber den anderen beiden aus), so definiert folgendes System ein einfaches Basisexperiment zu einem Reaktor, der nur drei verschiedene Molekülsorten umfasst:

$$\boxed{R_{\text{Basis}} = \{(s_1 + s_2 \longrightarrow s_o)\} \qquad S_{\text{Basis}} = \{s_1, s_2, s_o\}} \tag{4.3}$$

Im Anschluss folgt eine Analyse dieses Systems, in dem Molekül s_o durch Reaktion der Edukte s_1 und s_2 produziert wird. Variiert wird die Ersetzungsmethode und damit der Algorithmus A_{Basis}.

Die geringe Komplexität der Basisreaktion 4.3 ermöglicht das Aufstellen von nicht-linearen Differenzialgleichungen sowie deren mathematisch exakte Lösung. Bei der Eduktersetzung ersetzt s_o mit gleicher Wahrscheinlichkeit s_1 oder s_2. Die freie Ersetzung dagegen definiert eine *Überschussproduktion*. Da die Zahl der Moleküle im Reaktor konstant ist, müssen beliebige Moleküle verdrängt werden. Die Abfuhr dieser „Abfallprodukte“ wird durch Abflussfunktion $\Phi(t)$ gesteuert, von Hofbauer und Sigmund (1984) auch *Verdünnungsfluss* genannt. $\Phi(t)$ wird gerade so angesetzt, dass die Summe aller Konzentrationsänderungen zu jeder Zeit Null ergibt. Diese Abflussfunktion wirkt auf alle Moleküle des Reaktors gleichermaßen (ständiges Rühren sorgt für räumlich homogene Verhältnisse im Reaktor), also auf jede Molekülsorte proportional zu ihrer Konzentration. Bezeichnet man die Konzentration eines Moleküles X zum Zeitpunkt t mit $|X|_t,\ 0 \leq |X|_t \leq 1$, so ergeben sich nach (2.4)[1] die folgenden Differenzialgleichungssysteme für die freie Ersetzung (indiziert durch Symbol f) bzw. für die Eduktersetzung (Index e):

Dgls. zur Freiersetzung:

$$\frac{d}{dt}|s_1|_t^f = -|s_1|_t^f * \Phi(t) \tag{4.4}$$

$$\frac{d}{dt}|s_2|_t^f = -|s_2|_t^f * \Phi(t) \tag{4.5}$$

$$\frac{d}{dt}|s_o|_t^f = 2 * |s_1|_t^f * |s_2|_t^f - |s_o|_t^f * \Phi(t) \tag{4.6}$$

$$\Phi(t) = 2 * |s_1|_t^f * |s_2|_t^f$$

[1] Es ist dabei zu beachten, dass (4.1) und (4.2) jeweils in 2 (gleiche) Regeln umzusetzen sind, da die Edukte in der Reihenfolge $s_1 \times s_2$ oder $s_2 \times s_1$ kollidieren. Wie erwähnt verläuft die Bildung des Produkts aber unabhängig davon.

Dgls. zur Eduktersetzung:

$$\frac{d}{dt}|s_1|_t^e = -|s_1|_t^e * |s_2|_t^e \tag{4.7}$$

$$\frac{d}{dt}|s_2|_t^e = -|s_1|_t^e * |s_2|_t^e \tag{4.8}$$

$$\frac{d}{dt}|s_o|_t^e = 2 * |s_1|_t^e * |s_2|_t^e \tag{4.9}$$

Neben $t \geq 0$ und $|s_o|_t^{\{e,f\}} + |s_1|_t^{\{e,f\}} + |s_2|_t^{\{e,f\}} = 1$ gilt dabei jeweils $0 \leq |X|_t^{\{e,f\}} \leq 1$ für alle Molekülspezies X sowohl unter Eduktersetzung als auch unter Freiersetzung. Im Falle der freien Ersetzung ergibt sich der Abfluss Φ aus der Summe aller neu gebildeten Moleküle, hier: Zahl neuer s_o-Moleküle. Spezifiziert man die Anfangsbedingungen, also die Konzentration aller Moleküle zu Anfang des Experiments, so können die Differenzialgleichungssysteme exakt gelöst werden.

4.1.1 Lösung der Differenzialgleichungssysteme

Die Anfangsbedingungen für das Basisexperiment für geschlossene Reaktoren lassen sich durch Parameter a und b wie folgt festlegen:

$$(|s_1|_0, |s_2|_0, |s_o|_0) = (a, b, 1-a-b)$$

Folgende Bedingungen müssen dabei gelten: $a, b \geq 0$, $a+b \leq 1$, Für $t \geq 0$ und mit $\delta_{ab} = a - b$ können die Konzentrationen der drei Molekülsorten durch analytische Lösung des Differenzialgleichungssystems angegeben werden:

$$(|s_1|_t^f, |s_2|_t^f, |s_0|_t^f) = \left(\frac{a}{\sqrt{4abt+1}}, \frac{b}{\sqrt{4abt+1}}, \frac{\sqrt{4abt+1}-a-b}{\sqrt{4abt+1}} \right) \tag{4.10}$$

$$(|s_1|_t^e, |s_2|_t^e, |s_0|_t^e) = \begin{cases} \frac{a}{ta+1}, \frac{a}{ta+1}, \frac{ta+1-2a}{ta+1} & a = b \\ \frac{\delta_{ab} a e^{(\delta_{ab} t)}}{a e^{(\delta_{ab} t)} - b}, \frac{\delta_{ab} b}{a e^{(\delta_{ab} t)} - b}, 1 - |s_1|_t^e - |s_2|_t^e & a \neq b \end{cases} \tag{4.11}$$

Definition 4.1.1 (Aktive und passive Moleküle, Charakterisierung der Anfangswerte)

Während Moleküle des Typs s_1 und s_2 in produktiven Reaktionen als Edukt auftauchen können, so gilt dies nicht für solche des Typs s_o. Letztere werden daher als *passive* Moleküle bezeichnet, die Erstgenannten dagegen als *aktive*.

Weiterhin wird das Molekülgemisch, das zu Anfang des Experiments in den Reaktor gefüllt wird, *Initialbefüllung* genannt. Eine solche Initialbefüllung heißt *balanciert*, wenn alle verfüllten Molekültypen in der gleichen Konzentration im Reaktor vorliegen.

Der Grad der Auswirkung unbalancierter Initialbefüllungen auf die Konzentrationsverläufe der im Reaktor befindlichen Molekülsorten heißt *Anfangswert-Sensitivität* oder auch *Sensitivität gegenüber der Initialbefüllung*. Für eine vorgegebene Molekülsorte X und Ersetzungsvariante ε wird Sensibilität α definiert durch:

$$\alpha_X^{\varepsilon}(t) = \max_{\substack{a,b\geq 0 \\ a+b\leq 1}} |X|_t^{\varepsilon} - \min_{\substack{a,b\geq 0 \\ a+b\leq 1}} |X|_t^{\varepsilon}. \tag{4.12}$$

Diese Definition wird bei Zuflussreaktoren auf natürliche Weise erweitert zu $\alpha_X^{\varepsilon}(t,\Psi)$.

Bei einer geringen Anfangswert-Sensitivität ist der Einfluss der Parameter a und b auf die Molekülkonzentrationen nach einer gewissen Einschwingphase eher gering. Werden die Konzentrationen dagegen stark von der Wahl von a und b beeinflusst, reagiert der Reaktor stark sensitiv gegenüber der Initialbefüllung. In der anschließenden Analyse wird zwischen balancierten und unbalancierten Initialbefüllungen unterschieden, des Weiteren, ob diese Initialbefüllungen nur aktive oder auch passive Moleküle enthalten.

4.1.2 Balancierte Initialbefüllung mit aktiven Molekülen

Betrachtet man zunächst den einfacheren Fall, dass sich eingangs des Experiments jeweils gleich viele Moleküle der Sorten s_1 und s_2 im Reaktor befinden ($a = \frac{1}{2}, b = \frac{1}{2}$), so ist im Langzeitverhalten Übereinstimmung der Ersetzungsvarianten bezüglich der Grenzwerte der Konzentrationen aller beteiligten Moleküle festzustellen (Übergang $t \to \infty$).

$$\lim_{\substack{t\to\infty \\ a=b=\frac{1}{2}}} |s_o|_t^{\{e,f\}} = 1 \tag{4.13}$$

$$\lim_{\substack{t\to\infty \\ a=b=\frac{1}{2}}} |s_1|_t^{\{e,f\}} = 0 \tag{4.14}$$

$$\lim_{\substack{t\to\infty \\ a=b=\frac{1}{2}}} |s_2|_t^{\{e,f\}} = 0 \tag{4.15}$$

s_1 und s_2 werden komplett und – bezogen auf die gleiche Ersetzungsvariante – gleichschnell verbraucht, um s_o zu produzieren. Abbildung 4.1 zeigt den Vergleich zwischen den beiden Varianten. Es sind unterschiedliche Konvergenzgeschwindigkeiten zu beobachten: Eduktersetzung konvergiert deutlich schneller als die freie Variante. Nach $t = \frac{5}{4}$ sind bei Letztgenannter die Konzentrationen aller Molekülsorten gleich hoch, sie betragen $\frac{1}{3}$. Dieser „Isozustand“ tritt bei der Eduktersetzung schon bei $t = 1$ auf.

4.1.3 Unbalancierte Initialbefüllung mit aktiven Molekülen

Bei anfänglicher ungleicher Befüllung ($a \neq \frac{1}{2}$, $b = 1 - a$) treten deutlichere Unterschiede zu Tage. Die Gleichungen 4.13 bis 4.15 können für diesen Fall nur teilweise übernommen werden wie Abb. 4.2 zeigt. Ist nämlich Eduktersetzung gewählt, kann die Konzentration der Sorte s_o höchstens auf das Zweifache der Konzentration der initial niedriger konzentrierten Molekülsorte anwachsen.

Weiterhin geht aus Abb. 4.2 hervor, dass diese Aussage nicht auf die andere, die freie Ersetzungsvariante übertragen werden kann. Sofern überhaupt Moleküle der Sorten s_1 und s_2 vorhanden waren, wird nach ausreichender Zeit der Reaktor fast vollständig mit solchen des Typs s_o gefüllt sein. Obwohl die niedriger konzentrierte Sorte in der freien Ersetzung in keiner Weise vor dem Aussterben geschützt ist, wird sie nicht wie bei der Eduktersetzung (vorzeitig) verdrängt, denn für sie wird es mit dem fortschreitenden Absinken ihrer Konzentration immer unwahrscheinlicher, tatsächlich ersetzt zu werden. Auf diese Weise kann eine Molekülsorte bei noch so geringer Kon-

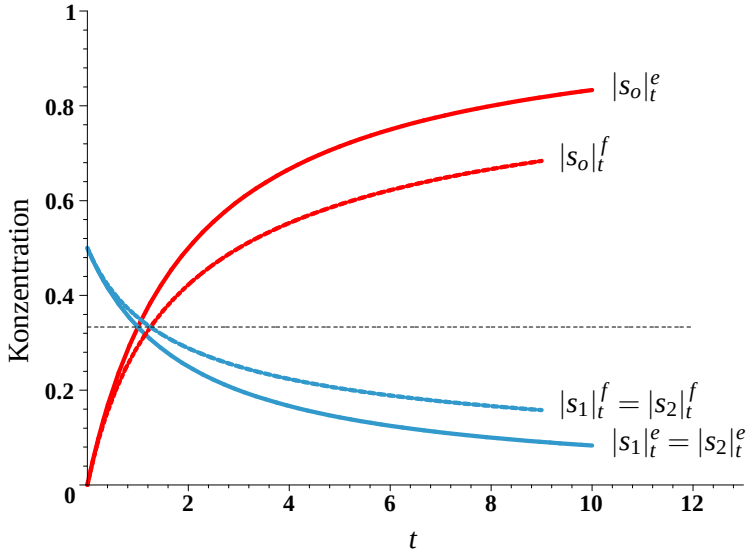


Abbildung 4.1: Konzentrationsverlauf der Moleküle s_1, s_2 und s_o im Basisexperiment (gegeben durch Gleichung 4.3) mit $a = b = \frac{1}{2}$. Gegenüberstellung von Frei- und Eduktersetzung bei ausgeglichener Initialbefüllung mit Molekülen der Sorten s_1 und s_2. Für gleiche Molekülsorten gelten gleiche Grenzwerte unter beiden Ersetzungsstrategien. Es sind jedoch unterschiedliche Konvergenzgeschwindigkeiten zu beobachten. Die strichlierte Linie zeigt das Konzentrationsniveau $\frac{1}{3}$ an.

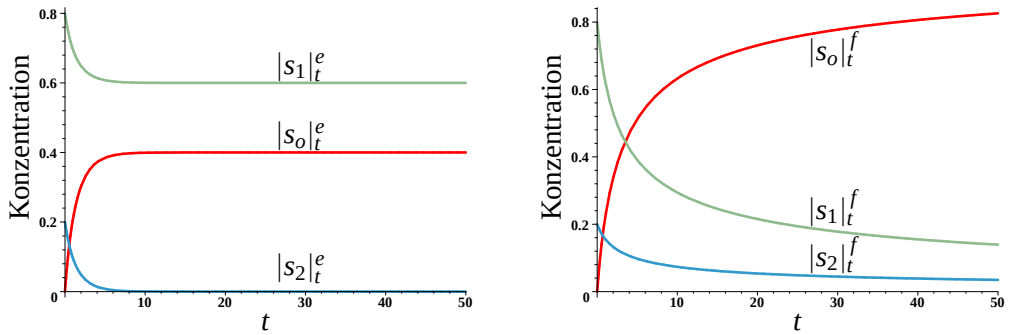


Abbildung 4.2: Konzentrationsverlauf der Moleküle s_1, s_2 und s_o im Basisexperiment (gegeben durch Gleichung 4.3) mit unbalancierter Initialbefüllung und geschlossenem Reaktor. Vergleich von Eduktersetzung (links) und Freiersetzung (rechts) für $a = 0.8$, $b = 0.2$. Deutlich unterschiedliches Verhalten wird sichtbar. Erklärung siehe Text.

zentration – genügend Zeit vorausgesetzt – zur Produktion von s_o genutzt werden. Eduktersetzung dagegen bietet in jeder Reaktion für beide Edukte konzentrationsunabhängig eine 50-prozentige Ersetzungswahrscheinlichkeit. So gilt für $0 \neq a \neq 1$ und $\delta_a = a - b = a - (1-a) = 2a - 1$:

$$\lim_{t\to\infty} |s_o|_t^e = 1 - |\delta_a| \qquad \lim_{t\to\infty} |s_o|_t^f = 1 \tag{4.16}$$

$$\lim_{t\to\infty} |s_1|_t^e = \max(\delta_a, 0) \qquad \lim_{t\to\infty} |s_1|_t^f = 0 \tag{4.17}$$

$$\lim_{t\to\infty} |s_2|_t^e = \max(-\delta_a, 0) \qquad \lim_{t\to\infty} |s_2|_t^f = 0 \tag{4.18}$$

Abbildungen 4.3 und 4.4 zeigen die dynamische Entwicklung der Konzentrationen über den gesamten Definitionsbereich der $s_i, i \in \{1,2,o\}$. Im Zentrum des durch $(|s_1|, |s_2|)$ aufgespannten Simplex ist das Verhalten der beiden Ersetzungsvarianten qualitativ ähnlich. Je mehr man sich

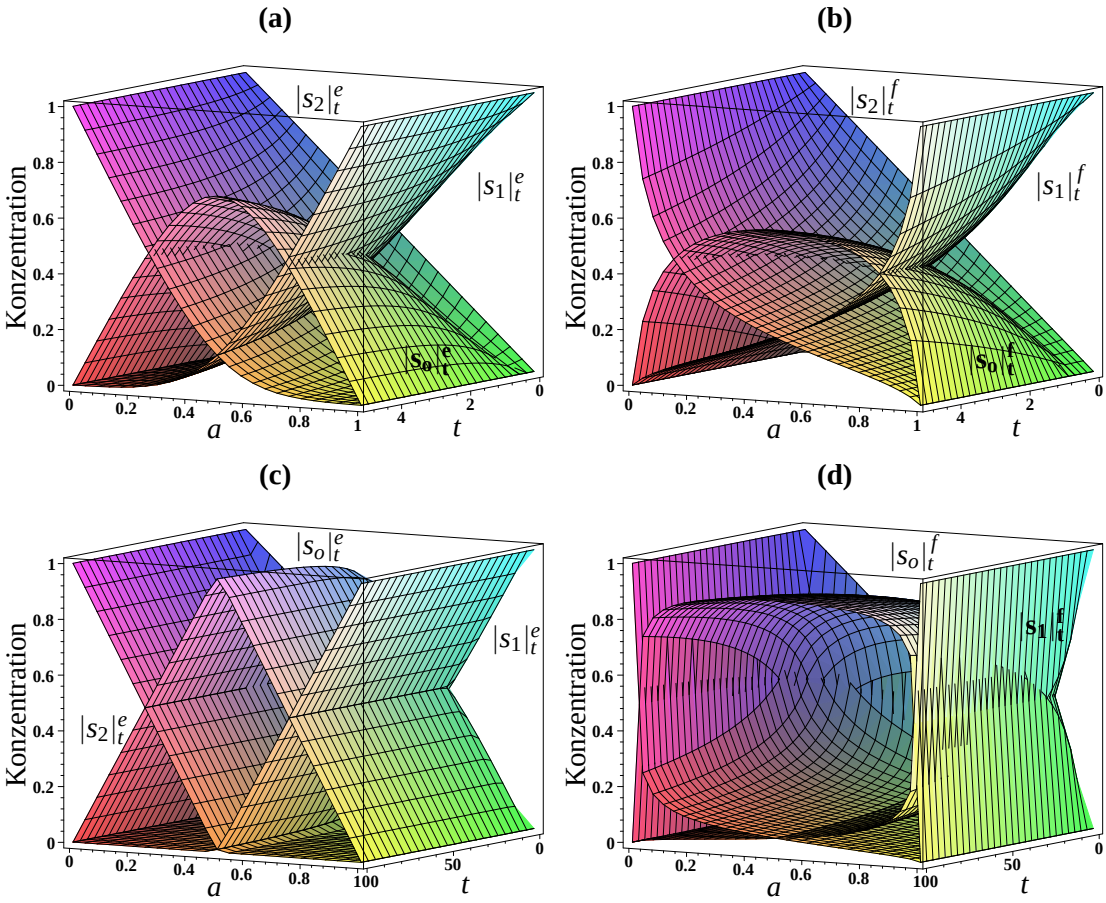


Abbildung 4.3: Die beiden Ersetzungsvarianten von *RESAC* im Basisexperiment bei geschlossenem Reaktor. Der Konzentrationsverlauf der Molekültypen s_1, s_2 und s_o ist aufgetragen über die Zeit und die Anfangskonzentration a der Molekülsorte s_1. Die Startkonzentration von s_2 beträgt $1-a$, s_o ist anfangs nicht vertreten. Mit jeweils zwei Zeitskalen wird die Eduktersetzung in (a) und (c) der Freiersetzung in (b) und (d) gegenübergestellt. Vergleiche dazu Abb. 4.4, weitere Erklärung siehe Text.

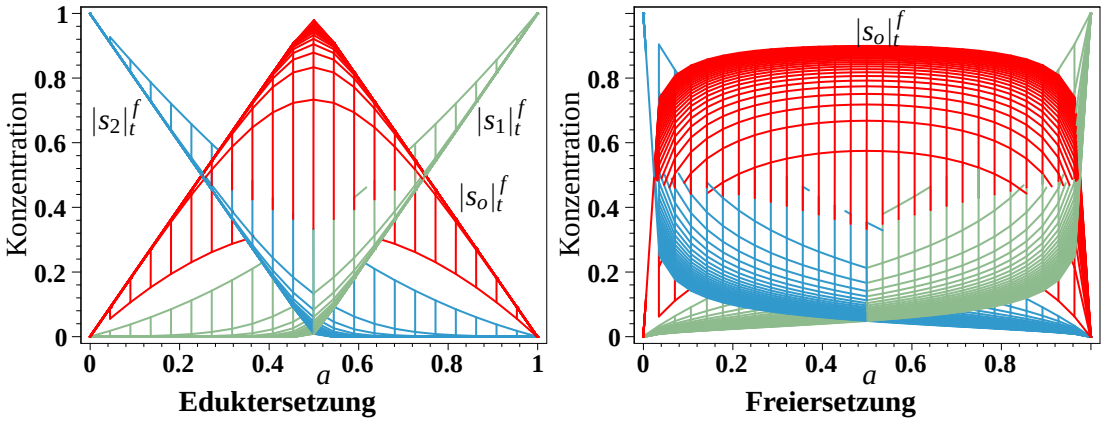


Abbildung 4.4: Die Graphen zeigen die Projektionen der in Abb. 4.3 dargestellten Flächenverläufe auf die $(t = 100)$-Ebene. Deutlich wird hier die mit der Zeit fortschreitende Ausbildung einer Spitze im s_o-Konzentrationsverlauf unter Eduktersetzung (links) bei $a = \frac{1}{2}$, also bei balancierter Initialbefüllung. In Richtung $a = 0$ und $a = 1$ nimmt die Konzentration dann linear ab. Mit der freien Ersetzungsvariante (rechts) hingegen ist die Maximalkonzentration von s_o fast über den ganzen Bereich des Parameters a maximal. Erst zu den Seiten hin flacht der Verlauf ab. Nur bei $a = 0$ und $a = 1$ kann kein s_o-Molekül mehr erzeugt werden. Zu beachten ist wieder der Unterschied in den Maximalkonzentrationen von s_o. Konvergenz wird unter der freien Ersetzung langsamer erreicht. Andererseits gilt der in Gleichung 4.16 angegebene Grenzwert.

zum Rand dieses Simplex begibt, desto stärker ist die Diskrepanz zwischen den Varianten. Im Gegensatz zur freien Ersetzung verliert dort die Eduktersetzung mehr und mehr die Fähigkeit, s_o-Moleküle zu produzieren.

Definiert man $\Delta_i(t) = |s_i|_t^e - |s_i|_t^f$ für $i \in \{1, 2, o\}$, also ein Maß für die Abweichung der freien Ersetzung zur Eduktersetzung, so wird dieser Sachverhalt noch einmal sehr deutlich (Abb. 4.5). Der durch $\Delta_i(t)$ gegebene Fehler ist ein durch Überlagerung zweier verschiedener Fehlerursachen resultierender Gesamtfehler. Wie im letzten Unterkapitel 4.1.2 gesehen, sind die Grenzwerte in der Simplexmitte gleich, in der näheren Umgebung ähnlich. Der zu beobachtende Fehler resultiert aus den unterschiedlichen Konvergenzgeschwindigkeiten und wird daher *Konvergenzfehler* genannt. Der Betrag des Konvergenzfehlers nimmt mit zunehmend verstreichender Experimentalzeit t ab und strebt für $a = \frac{1}{2}$ gegen Null.

Ist der Reaktor hingegen unbalanciert befüllt worden, gibt es den schon begründeten qualitativen Unterschied. Dieser Fehler, *Strukturfehler* genannt, nimmt mit dem Grade der Unbalanciertheit zu. Auch bei fortschreitender Zeit nimmt der Betrag des Strukturfehlers niemals ab, sondern ist monoton steigend.

4.1.4 Unbalancierte Initialbefüllung mit aktiven und passiven Molekülen

Zum Dritten wird der Fall einer unbalancierten Initialbefüllung betrachtet, in dem nun auch Moleküle des Typs s_o von Beginn an im Reaktor vertreten sein können. Es handelt sich um die allgemeinste Formulierung der Anfangsbedingungen zum Basisexperiment geschlossener Reaktoren.

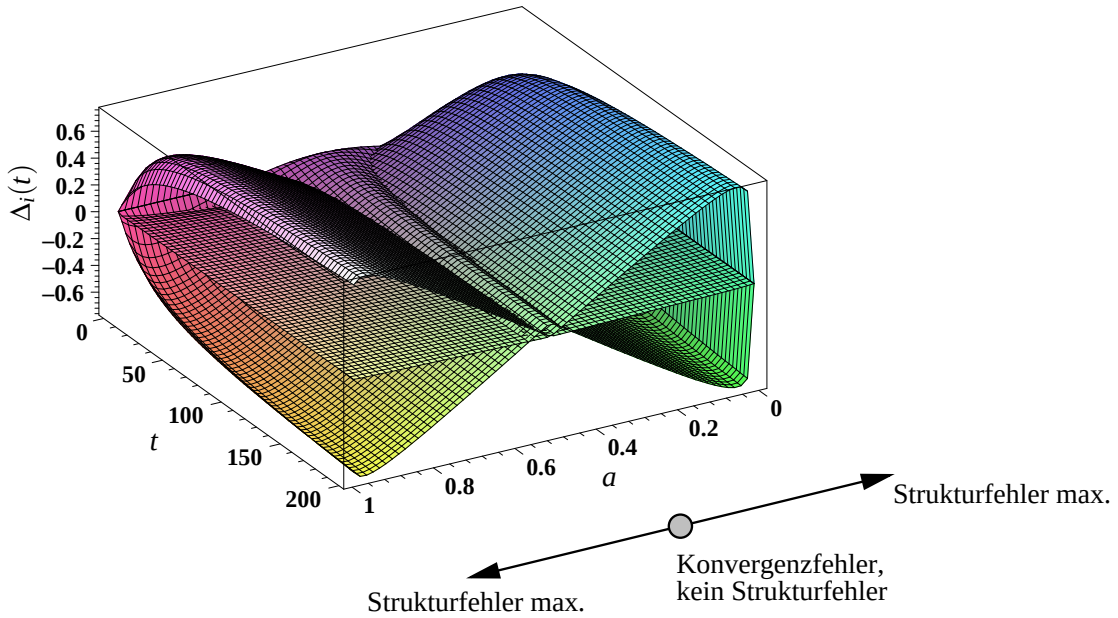


Abbildung 4.5: Abweichung der freien Ersetzung von der Eduktersetzung in geschlossenen Reaktoren. $\Delta_i(t)$ gibt die Abweichung für alle drei Molekültypen an, $i \in \{1,2,o\}$. Zwei verschiedene Fehlertypen sind unterscheidbar: Der mit der Zeit abnehmende Konvergenzfehler bei annähernd balancierter Initialbefüllung sowie der bei unbalancierte Befüllung auftretende Strukturfehler.

Wählt man $\delta_{ab} = a - b$, so ergeben sich aus den Konzentrationsfunktionen 4.10 und 4.11 für $a, b > 0$ und $a + b \leq 1$ die folgenden Grenzwerte:

$$\lim_{t\to\infty} |s_o|_t^e = 1 - |\delta_{ab}| \qquad \lim_{t\to\infty} |s_o|_t^f = 1 \tag{4.19}$$

$$\lim_{t\to\infty} |s_1|_t^e = \max(\delta_{ab}, 0) \qquad \lim_{t\to\infty} |s_1|_t^f = 0 \tag{4.20}$$

$$\lim_{t\to\infty} |s_2|_t^e = \max(-\delta_{ab}, 0) \qquad \lim_{t\to\infty} |s_2|_t^f = 0 \tag{4.21}$$

Ist $a = 0$ oder $b = 0$, so gilt $\lim_{t\to\infty} |s_o|_t^{\{e,f\}} = 1 - a - b$.

Das Zulassen passiver Moleküle in der Initialbefüllung führt erneut zum schon erwähnten Strukturfehler. s_o-Moleküle können in der Eduktersetzung nie ersetzt werden, da sie niemals Edukte einer produktiven Reaktion sein können. Das Ersetzen von Edukten durch Produkte wirkt bezogen auf die Gesamtpopulation im Reaktor *lokal*, d. h. nur auf die an der jeweiligen produktiven Reaktion beteiligten Moleküle. Dieses lokale Verhalten kann zur *Nischenbildung* führen: passive Moleküle unterliegen keinem Konkurrenzdruck.

In diesem Sinne *global* wirkt demgegenüber die freie Ersetzung auf sämtliche Moleküle des Reaktors. Selbst relativ kleine Spuren aktiver Moleküle reichen aus, um eine „volle Reaktionsantwort“ zu induzieren, hier: hohe s_o-Produktion. Solch niedrig konzentrierte Moleküle, die umfangreiche Reaktionen auslösen können, werden *Starter* genannt. Nischenbildung kann es nicht geben, da der Ersetzungsdruck auf alle Moleküle in gleichem Maße wirkt.

Eine andere Folge der globalen Wirkung der Freiersetzung ist die schon in den beiden vorhergehenden Unterkapiteln angesprochene niedrigere Konvergenzgeschwindigkeit. Auch die gewünschten Produkte, in diesem Falle s_o-Moleküle, werden durch produktive Reaktionen ersetzt. Und zwar mit einer zu ihrer Konzentration proportionalen Wahrscheinlichkeit.

4.2 Das Basisexperiment für Zuflussreaktoren

Geschlossene Reaktoren weisen eine starke Abhängigkeit von der Initialbefüllung auf, wenn das Schema der Eduktersetzung angewandt wird. Ob sich offene Architekturen mit geregeltem Molekülzufluss anders verhalten und welche Eigenschaften solche Architekturen aufweisen, wird in diesem Unterkapitel untersucht. Es wird im Folgenden von einem Zufluss ausgegangen, der zu gleich vielen Anteilen aus allen aktiven Molekülen besteht. Im Basisexperiment bedeutet dies einen Zufluss von s_1 und s_2 zu gleichen Teilen. Die Zuflussrate Ψ, $0 < \Psi \leq 1$, bezeichnet dabei einen Zufluss bei jeder $\frac{1}{\Psi}$-ten Kollision. Ein 50%-Zufluss ($\Psi = 0.5$) kennzeichnet so einen Zufluss bei jeder zweiten Kollision unabhängig davon, ob diese Kollision eine produktive Reaktion zur Folge hat oder nicht. In einer Generation werden einem Reaktor der Größe $|M|$ so $\Psi * |M|$ neue Moleküle zugeführt. Der Zufluss verdrängt die Sorten im Reaktor proportional zu ihren Konzentrationen. Die resultierenden Differenzialgleichungen für die freie Ersetzung bzw. für die Eduktersetzung lauten:

Dgls. zur Freiersetzung:

$$\overbrace{\phantom{\frac{d}{dt}|s_1|_t^f =}}^{*}\ \overbrace{\phantom{\frac{\Psi}{2}}}^{**}\ \overbrace{}^{***}$$

$$\frac{d}{dt}|s_1|_t^f = \frac{\Psi}{2} - |s_1|_t^f * \Phi(t) \tag{4.22}$$

$$\frac{d}{dt}|s_2|_t^f = \frac{\Psi}{2} - |s_2|_t^f * \Phi(t) \tag{4.23}$$

$$\frac{d}{dt}|s_o|_t^f = 2 * |s_1|_t^f * |s_2|_t^f - |s_o|_t^f * \Phi(t) \tag{4.24}$$

$$\Phi(t) = 2 * |s_1|_t^f * |s_2|_t^f + \Psi$$

Dgls. zur Eduktersetzung:

$$\overbrace{\phantom{\frac{d}{dt}|s_1|_t^e = -|s_1|_t^e * |s_2|_t^e +}}^{*}\ \overbrace{\phantom{\frac{\Psi}{2}}}^{**}\ \overbrace{}^{***}$$

$$\frac{d}{dt}|s_1|_t^e = -|s_1|_t^e * |s_2|_t^e + \frac{\Psi}{2} - |s_1|_t^e * \Psi \tag{4.25}$$

$$\frac{d}{dt}|s_2|_t^e = -|s_1|_t^e * |s_2|_t^e + \frac{\Psi}{2} - |s_2|_t^e * \Psi \tag{4.26}$$

$$\frac{d}{dt}|s_o|_t^e = 2 * |s_1|_t^e * |s_2|_t^e - |s_o|_t^e * \Psi \tag{4.27}$$

Neben $t \geq 0$ gilt dabei jeweils $0 \leq |X|_t^{\{e,f\}} \leq 1$ für alle Molekülspezies X. $\Phi(t)$ kennzeichnet im Falle der freien Ersetzung den Abfluss aus dem Reaktor und ist gerade so gewählt, dass sich die Konzentrationsänderungen in der Summe aufheben: $\sum_{i=\{1,2,o\}} \frac{d}{dt}|s_i|_t^{\{e,f\}} = 0$. Die Formeln bestehen jeweils aus drei Teilen. Der erste Teil (*) gibt die Abnahme (im Falle von s_1 und s_2) bzw. die Zunahme (bei Sorte s_o) durch Reaktionen des Typs $s_1 \times s_2$ an. Durch den zweiten Formelabschnitt (**) wird die Zunahme der Molekülsorte durch den Zufluss Ψ beziffert: Er besteht zu jeweils 50% aus s_1- und s_2-Molekülen. Schließlich beschreibt der letzte Teil (***) die Verdrängung durch den Abfluss Φ. Bei der Eduktersetzung besteht der Abfluss nur aus der durch die einfließenden Moleküle verursachten Verdrängung.

In Gegensatz zum Basisexperiment für geschlossene Reaktor ist es bei Zuflussreaktoren nicht gelungen, geschlossene analytische Lösungen zu den Differenzialgleichungen 4.22 bis 4.27 zu bestimmen.

4.2.1 Gleichgewichtszustände

Ein Gleichgewichtszustand ist erreicht genau dann, wenn gilt:

$$\left(\frac{d}{dt}|s_1|_t^{\{e,f\}}, \frac{d}{dt}|s_2|_t^{\{e,f\}}, \frac{d}{dt}|s_o|_t^{\{e,f\}}\right) = (0,0,0). \tag{4.28}$$

Es bezeichne t_{eq} den Zeitpunkt, zu dem sich das System im Gleichgewicht befindet. Es gelte im Folgenden $\Psi > 0$.

Eduktersetzung: Aus Gleichungen 4.25 und 4.26 folgt zusammen mit (4.28) $|s_1|_{t_{eq}}^e = |s_2|_{t_{eq}}^e$. Mit $|s_o|_t^e = 1 - (|s_1|_t^e + |s_2|_t^e) = 1 - 2 * |s_1|_t^e$ und Gleichung 4.27 errechnet man

$$|s_1|_{t_{eq}}^e = -\frac{\Psi}{2} + \frac{\sqrt{\Psi(\Psi+2)}}{2} \tag{4.29}$$

$$|s_2|_{t_{eq}}^e = -\frac{\Psi}{2} + \frac{\sqrt{\Psi(\Psi+2)}}{2} \tag{4.30}$$

$$|s_o|_{t_{eq}}^e = 1 + \Psi - \sqrt{\Psi(\Psi+2)} \tag{4.31}$$

Der „Isozustand“, in dem die Konzentrationen aller 3 Molekülsorten gleich hoch sind, wird für $\Psi = \frac{2}{3}$ erreicht.

Freiersetzung: Aus Gleichungen 4.22 und 4.23 folgt mit Gleichung 4.28 erneut $|s_1|_{t_{eq}}^f = |s_2|_{t_{eq}}^f$. Auflösen des Gleichungssystems liefert:

$$|s_i|_{t_{eq}}^f = -\frac{\sqrt[3]{3}}{6} \frac{\left(-\Psi^{\frac{2}{3}}\sigma_\Psi^2 + 2\Psi\sqrt[3]{3}\right)}{\sqrt[3]{\Psi}\sigma_\Psi} \text{ für } i \in \{1,2\} \tag{4.32}$$

$$|s_o|_{t_{eq}}^f = 1 + \sqrt[3]{\frac{1}{9}} \frac{\left(-\Psi^{\frac{2}{3}}\sigma_\Psi^2 + 2\Psi\sqrt[3]{3}\right)}{\sqrt[3]{\Psi}\sigma_\Psi} \tag{4.33}$$

$$\sigma_\Psi = \sqrt[3]{9 + \sqrt{3}\sqrt{8\Psi + 27}}$$

Aus den Gleichungen ergibt sich, dass für einen Zufluss von $\Psi = \frac{4}{9}$ alle Molekülsorten gleich konzentriert sind.

Bei der Betrachtung der Gleichgewichtszustände beider Ersetzungsstrategien fällt auf, dass sie unabhängig von den Parametern a und b sind, also anfangswertunabhängig sind. In Unterkapitel 4.2.3 wird näher darauf eingegangen. Numerische Integration zeigt, dass diese Zustände tatsächlich angestrebt werden.

4.2.2 Reaktorverhalten bei Vergrößerung des Zuflusses

Die Formeln 4.29 bis 4.31 und 4.32 bis 4.33 lassen drei grundsätzlich verschiedene Konzentrationsbilder in offenen Reaktoren unter variierendem Zufluss erkennen. Die folgende Klassifikation gilt qualitativ sowohl für Frei- als auch für die Eduktersetzung:

$|s_1|_t \neq |s_2|_t \neq |s_o|_t$ Ein solcher Zustand muss nicht notwendigerweise erreicht werden. Wenn er erreicht wird, dann ist das der Fall entweder in geschlossenen Reaktoren ($\Psi = 0$) oder in Reaktoren mit geringem Zufluss oder in Reaktoren mit nicht geringem Zufluss zu frühen Zeitpunkten t.

$|s_1|_t = |s_2|_t$ Dieser durch gleiche Konzentration aktiver Moleküle gekennzeichnete Zustand tritt für jeden Zufluss $\Psi > 0$ zu irgendeinem Zeitpunkt t ein. Wie oben gezeigt stimmen die Formeln für die Konzentrationen von s_1 und s_2 im Grenzwert überein. Siehe dazu auch Unterkapitel 4.2.3.

$|s_1|_t = |s_2|_t = |s_o|_t$ Gleichsetzen der Formeln 4.29 und 4.31 beziehungsweise 4.32 und 4.33 liefert den Zufluss, der ab einem bestimmten Zeitpunkt t zum „Isozustand“ führt, also zu gleichhohen Konzentrationen aller Molekülsorten. Für die Eduktersetzung ergibt sich ein Zufluss von $\Psi^e_{\text{iso}} = \frac{2}{3}$, für die freie Ersetzung ein Zufluss von $\Psi^f_{\text{iso}} = \frac{4}{9}$.

$\Psi < \Psi_{\text{iso}}$ Für Zuflüsse kleiner Ψ_{iso} gilt, dass die Molekülsorten s_1 und s_2 geringer konzentriert sind als s_o.

$\Psi > \Psi_{\text{iso}}$ Für Zuflüsse größer Ψ_{iso} gilt, dass die Molekülsorten s_1 und s_2 höher konzentriert sind als s_o.

Bewiesen wurden die Aussagen durch Analyse der Gleichgewichtszustände. Tatsächlich gelten die Aussagen angenähert schon zu relativ frühen Zeitpunkten t_{eq} wie in Abb. 4.6 zu sehen ist – die Abbildung entstand durch numerische Integration mittels des Runge-Kutta-Verfahrens mit Schrittweitenanpassung (Grosche et al. 1995). Im nächsten Unterkapitel wird eine Schätzung des Zeitpunkts angegeben, zu dem das grenzwertige Verhalten nach obigen Formeln eingenommen wird.

4.2.3 Anfangswert-Sensitivität und Zeiteinfluss

Im letzten Unterkapitel wurden Aussagen $\mathcal{A}$ formuliert vom Typ „Es gibt einen Zeitpunkt t ($t = \infty$ eingeschlossen), so dass $\mathcal{A}$ gilt für alle $\Psi > 0$“. Numerische Integration zeigt, dass dieser Zeitpunkt schon relativ früh eintritt. In Abbildung 4.7 ist die Konzentration der passiven Sorte s_o im 10. Zeitschritt in Abhängigkeit des Zuflusses Ψ unter verschiedenen Initialbefüllungen aufgetragen. Für $\Psi < \Psi_K$ mit $\Psi_K \approx 0.3$ erscheint der Zuflussreaktor Anfangswert-sensitiv, und der Grad der Sensitivität steigt exponentiell mit der Abnahme des Zuflusses. Für $\Psi > \Psi_K$ gilt, dass eine hinreichend gute Annäherung an den anfangswertunabhängigen Grenzwert (Übergang $t \to \infty$) erreicht wird. Dieser ist analytisch gegeben durch Formel 4.31 für Eduktersetzung und Formel 4.33 für die freie Ersetzung. Der Konvergenzpunkt Ψ_K ist abhängig von der Zeit. Er verschiebt sich mit fortschreitender Zeit in Richtung eines geringer werdenden Zuflusses, und damit verschiebt sich die Anfangswert-Sensitivität ebenso (siehe Abb. 4.8, Seite 86). Die Einflüsse von Zeit und Zufluss auf die Konzentrationen im Reaktor nehmen jeweils exponentiell ab.

Die Anfangswert-Sensitivität $\alpha^e_{s_o}(t,\Psi)$ lässt sich im Falle der Eduktersetzung experimentell bestimmen zu:

$$\alpha^e_{s_o}(t,\Psi) \approx e^{-c\Psi t} \text{ für } \Psi, t > 0 \tag{4.34}$$

Dabei ist $c = \frac{7}{4}$ (vergleiche Abb. 4.7). Zu gegebener Sensitivität und Zuflussrate lässt sich entsprechend die Zeit ermitteln, die benötigt wird, diese zu erreichen:

$$t \approx \frac{\ln \alpha^e_{s_o}(t,\Psi)}{-c\Psi} \tag{4.35}$$

Beispielsweise muss man zirka 12 Zeitschritte warten, um in einem Zuflussreaktor mit 35%-igem Zufluss und Eduktersetzung eine Anfangswert-Sensitivität von $\frac{1}{1000}$ zu erreichen. Reaktoren mit freier Ersetzung verlieren ihre Sensitivität gegenüber der Initialbefüllung bei noch kleineren Zufluss-/Zeitwerten.

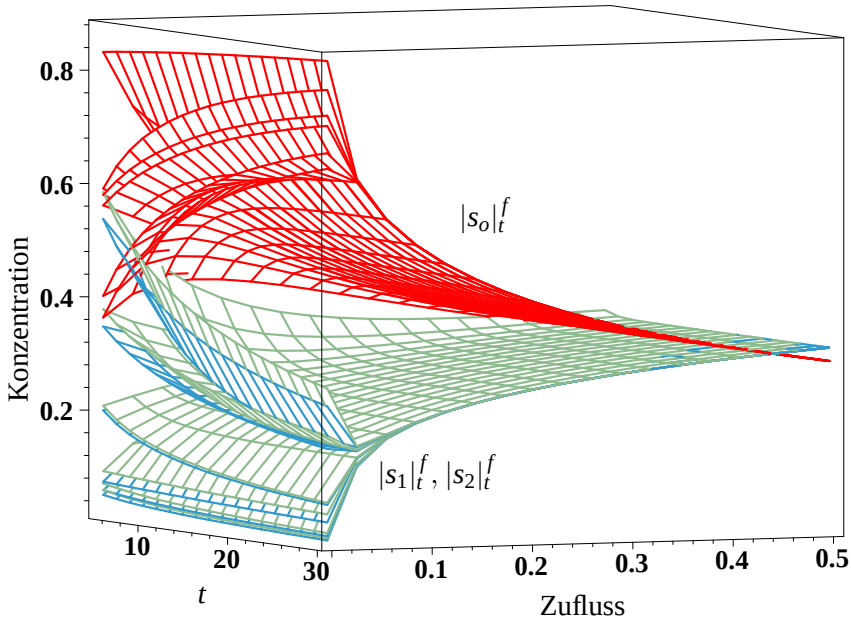


Abbildung 4.6: Konzentrationsverläufe im Basisexperiment unter freier Ersetzung und variierendem Zufluss. Das Bild zeigt die Konzentration der Molekülsorten s_1, s_2 und s_o für unbalancierte Initialbefüllungen. Die Kurvenverläufe für die Eduktersetzung sind qualitativ ähnlich. Charakterisierung siehe Text.

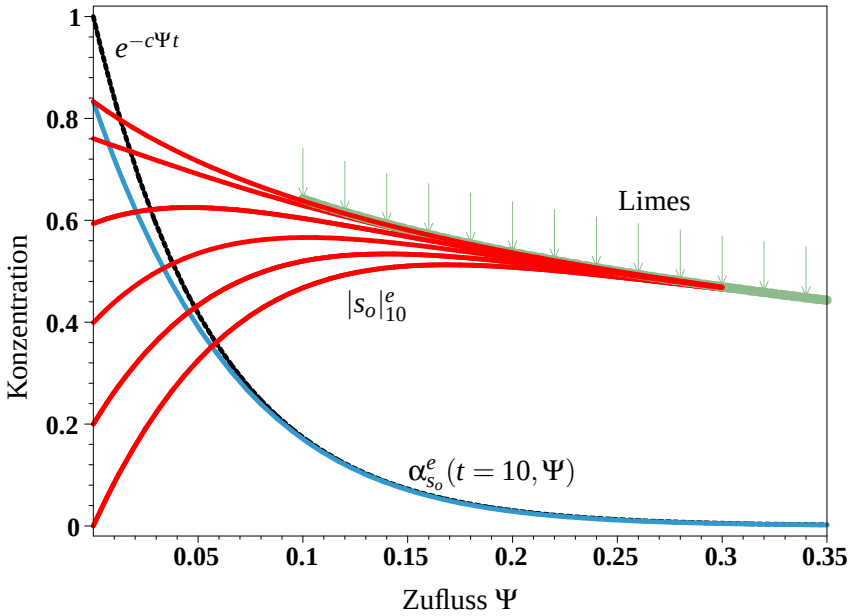


Abbildung 4.7: Einfluss des Zuflusses auf die Anfangswert-Sensitivität (hier: Eduktersetzung). Die zu unterschiedlichen Anfangswerten a und b gegebenen Konzentrationskurven von s_o vereinigen sich beim Konvergenzpunkt $\Psi_K \approx 0.3$ und folgen anschließend dem zeitunabhängigen Limes gemäß Gleichung 4.31. Die Abnahme der Sensitivität erfolgt exponentiell mit der Zeit und dem Zufluss. Er kann angenähert durch die Funktion $e^{-c\Psi t}$ mit $c = \frac{7}{4}$. Verhalten unter freier Ersetzung qualitativ ähnlich.

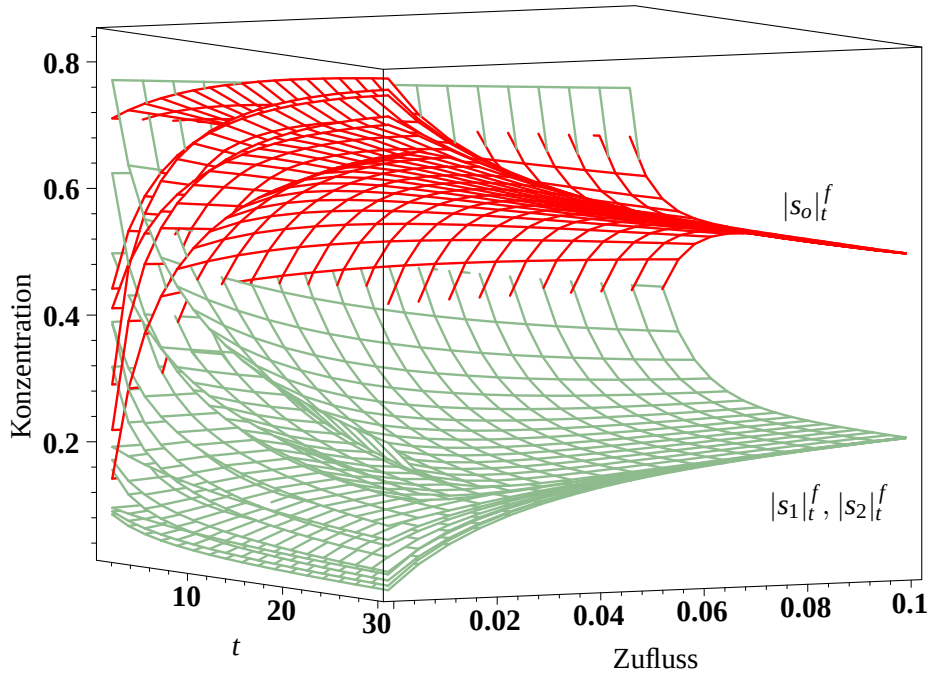


Abbildung 4.8: Konzentrationsverläufe im Basisexperiment in offenen Reaktoren unter freier Ersetzung in Abhängigkeit von Zeit und Zufluss. Mit zunehmender Zeit und zunehmenden Zufluss verschmelzen die durch unterschiedliche Initialbefüllungen hervorgerufenen unterschiedlichen Konzentrationskurven. Die Kurvenscharen der s_1- und s_2-Moleküle bündeln sich ebenso wie die der s_o-Moleküle. Die Anfangswert-Sensitivität nimmt schnell ab. Qualitativ ähnliches Verhalten findet man bei der Eduktersetzung.

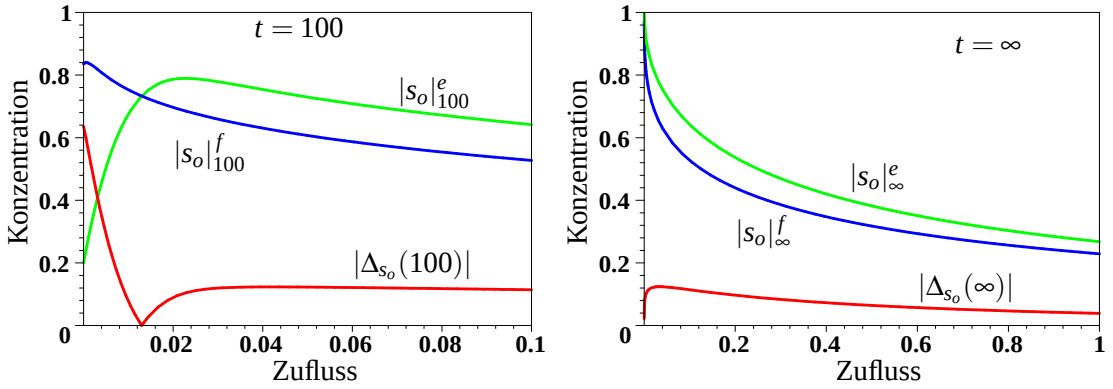


Abbildung 4.9: Frei- und Eduktersetzung in Zuflussreaktoren im Vergleich. *Links:* Konzentrationsverlauf im 100. Zeitschritt und unbalancierter Initialbefüllung. Unter Eduktersetzung profitiert man zunächst vom zunehmenden Zufluss und der Nachteil der ungleichmäßigen Befüllung kann gegenüber der freien Ersetzung wettgemacht und sogar ins Gegenteil verkehrt werden. Eine weitere Zunahme des Zuflusses wirkt sich auf beide Varianten negativ aus. *Rechts:* Situation zu fortgeschrittener Zeit.

4.2.4 Frei- versus Eduktersetzung: Einfluss des Zuflusses

Zufluss kann sich sowohl positiv als auch negativ auswirken. Es ist zwischen Frei- und Eduktersetzung zu differenzieren. Geringer Zufluss bringt Vorteile bei der Eduktersetzung, nämlich in dem Falle, dass keine balancierte Initialbefüllung garantiert werden kann. Ein aufgrund einer unbalancierten Befüllung unproduktiver Reaktor (hier im Sinne der Produktion von s_o) kann von einem Zufluss mit geringer Rate profitieren, seine Anfangswert-Sensitivität verlieren und seine Produktion erhöhen. Das Phänomen der Nischenbildung geht dabei verloren.

Andererseits hat eine hohe Zuflussrate eine Verschlechterung bezüglich der Konzentration von s_o zur Folge. Begründet ist dies in der mit zunehmenden Zufluss immer stärkeren Verdrängung des eigentlich gewünschten Produkts s_o durch die einfließenden Moleküle. Ganz markant macht dies die schon angesprochene Zuflussrate deutlich, bei der die Konzentrationen der aktiven Moleküle, die der passiven übersteigen. Abbildung 4.9 zeigt den Effekt der zunehmenden Verschlechterung. Außerdem zeigt sich die Eduktersetzung, wenn erst einmal unsensitiv gegenüber der Initialbefüllung, der freien Ersetzung überlegen. Der Grund liegt in ihrer lokalen Wirkung, die das passive Molekül s_o schützt (siehe auch Unterkapitel 4.1.4).

Eine Erhöhung der Zuflussrate über 1 hinaus, also der Zufluss von mehr als einem Molekül pro Kollision, bewirkt eine weitere Verringerung der s_o-Konzentration und führt in letzter Konsequenz erwartungsgemäß zur vollständigen Verdrängung der s_o-Moleküle. Sicherlich wird bei jeder produktiven Reaktion erneut ein solches Molekül hergestellt, doch durch den Zufluss schnell wieder verdrängt.

4.3 Einordnung der Thematik in den wissenschaftlichen Kontext

In Systemen, die der Analogie der Künstlichen Chemien folgen, also in irgendeiner Weise die Chemie als Vorbild haben, finden Reaktionen statt. Der Begriff der Reaktion ist dabei abstrakt zu sehen. Beispielsweise kann eine Reaktion auch die Austragung eines Wettbewerbs zweier Entitäten beschreiben. Eine solche Ausprägung als Turnier findet sich z. B. im System *Coreworld* (Rasmussen et al. 1990; Rasmussen et al. 1992), das auf der Architektur *Core War* (Dewdney 1984) aufbaut, einem so genannten Assembler-Automaten. Hier konkurrieren Computerprogramme auf einem Multiprozessorsystem mit gemeinsamen Speicherzugriff. Das von Ray (1992) entwickelte Modell *Tierra* ist ein weiteres Beispiel. Ebenso ist die Selbstreplikation als Reaktion einzustufen.

Reaktionen ordnen Edukten unter festgelegten Umständen Reaktionsprodukte zu. Diese Reaktionen werden üblicherweise, jedoch nicht zwingend, uni- oder bimolekular angelegt sein, was bedeutet, dass entweder ein oder zwei Edukte zur Reaktion notwendig sind. Allgemein formuliert allozieren die Edukte in ihrer Gesamtheit eine bestimmte Menge Platz[2]. Es stellt sich die Frage, wie mit den Produkten zu verfahren ist, *wo* diese zu platzieren sind. Wenn der akkumulierte Platz der Produkte sich von dem Platzbedarf der Edukte unterscheidet und Systeme konstanter Dichte modelliert werden, ist die Beantwortung dieser Frage darüber hinaus unerlässlich, da die

[2] Für diese Betrachtung ist dabei unerheblich, in welcher Form dies geschieht (beispielsweise linear oder durch Faltung in einer höheren Dimension).

Produkte die Edukte nicht auf intuitive Weise ersetzen können. Gleiches gilt für Systeme mit variabler Dichte, aber beschränkter Größe[3]. In solchen Systemen ist es auf Dauer unmöglich, alle in der Reaktion involvierten Moleküle zu behalten, ohne andere, unbeteiligte Moleküle zu entfernen. Information kann in geschlossenen Systemen der beschriebenen Art nicht unendlich wachsen. Es müssen also Reaktionen oder externe Mechanismen existieren, in Folge derer Information zerstört, d. h. überschrieben oder reorganisiert wird. Dafür gibt es mindestens vier sinnvolle Möglichkeiten. Die folgende Aufzählung erweitert die Darstellung in (Banzhaf 1995):

1. Eine beliebige, aber weder von den Erzeugern noch von dem Produkt getragene Information wird vernichtet. Die Produkte überschreiben an der Reaktion unbeteiligte Moleküle (Freiersetzung). Dabei werden entweder beliebige, zufällig gewählte Moleküle ersetzt (die Moleküle werden auf diese Weise proportional zu ihrer Konzentration selektiert), oder aber sie werden nach einem bestimmten Kriterium (Länge, Aufbau etc.) zur Ersetzung ausgewählt.

2. Die Information der/des direkten Erzeuger/s wird ausgelöscht. Produkte einer Reaktion überschreiben ganz oder teilweise die Edukte (Eduktersetzung).

3. Die durch die direkten Erzeuger gegebene Information wird reorganisiert. Produkte gehen durch Umordnung elementarer Bauteile aus den Edukten hervor.

4. Die Information sowohl der Edukte als auch der Produkte wird konserviert. Im Unterschied zu (1) wird ein vorübergehendes Informationswachstum toleriert. Zu bestimmten Zeitpunkten werden dann durch externe Mechanismen (z. B. fitnessbasierte Filterung) Informationen des Systems global ausgesiebt. Eine andere Möglichkeit ist das anfängliche Wachstum während einer Konstruktionsphase. Moleküle werden dem System nur solange zugefügt, wie ein beschränktes Reservoir an Grundbausteinen deren Konstruktion zulässt.

In endlichen Systemen wird die Beschränkung der Ressourcen früher oder später zu einem Wettstreit der Entitäten führen. (1) bis (4) definieren unterschiedliche Rahmenbedingungen für die Konkurrenzsituation. In jedem der vorgestellten Fälle wird das Produkt behalten.[4] Bei der freien Ersetzung wird – im Gegensatz zur Eduktersetzung – zudem auch die Erzeugerinformation konserviert. Die Perspektive einer Informationseinheit einnehmend, formuliert es Fontana (1992) so: Information wird nicht aufgebraucht durch ihre Benutzung. Durch den dritten Ansatz wird Information zwar nicht wirklich aufgebraucht, doch in einen neuen Kontext gestellt. Ansatz (4) vermag allein, zumindest in Phasen ungebremsten Wachstums die Konkurrenz zeitweise aufzuheben.

Durch (3) wird der Ansatz verfolgt, wie in der Chemie Reaktionsprodukte durch Auflösung von Bindungen zwischen Grundbausteinen der Edukte und anschließender Neuformation zu gewinnen. Kennzeichnend ist die ausgeglichene Materialbilanz einer jeden Reaktion. Systeme, die dieses Schema implementieren, wünschen meist eine relativ exakte Modellierung der Chemie wie in (Mayer und Rasmussen 1998; Bersini 2000; Benkö et al. 2003; Hutton 2003).

[3] Die Größe kann auf unterschiedliche Weise festgelegt sein, beispielsweise durch eine beschränkte Molekülzahl oder durch ein beschränktes Reservoir an Grundbausteinen, aus denen sich die Moleküle zusammensetzen.

[4] Das Produkt direkt zu verwerfen, nachdem es gebildet wurde, erscheint im Allgemeinen nicht nützlich zu sein. Abzugrenzen ist aber der Fall, dass eine Reaktion gar nicht erst zu Stande kommt, beispielsweise aufgrund nicht zur Verfügung stehender Grundbausteine.

Eine Abwandlung von (3) wird von Speroni di Fenizio (2000) beschrieben: Hier findet eine Art von *globaler* Reorganisation statt. Es gibt eine konstante Menge von Bausteinen in einem separaten *Reservoir*, jeder Bausteintyp tritt dort in einer festgelegten Anzahl auf. Bei Reaktionen werden die Bausteine der Edukte freigesetzt und dem Reservoir zugeführt werden. Sofern dies möglich ist, werden anschließend dem Reservoir Bausteine entnommen und zur Produktsynthese genutzt. Falls die zur Bildung des Reaktionsprodukts nötigen Bausteine nicht vorrätig sind, findet die Reaktion nicht statt.

Die Ersetzung *aller* Edukte ist typisch für Modelle, deren Reaktionsschemata die Grundidee so genannter Ersetzungssysteme[5], engl. rewrite systems, implementieren. Man kann hier von einer Art vollständigen Eduktersetzung sprechen, da im Gegensatz zu *RESAC* alle Edukte zerstört werden. Beispiele liefern (Berry und Boudol 1992; Kanada und Hirokawa 1994; Suzuki et al. 1996; Centler et al. 2003). Überproduktion (die Zahl der Produkte übersteigt die der Edukte) ist dann entweder nicht erlaubt oder führt zu einer Änderung der Molekülzahl, gegebenenfalls zu einer Dichteänderung.

Die teilweise Eduktersetzung – wie *RESAC* sie vorsieht – ist hingegen nicht weit verbreitet. Ein Beispiel für deren Einsatz ist die Dynamik der Primzahlreaktoren von Banzhaf et al. (1996) sowie Dittrich (1998). Hier wird im Gegensatz zu der vorgestellten Eduktersetzung in *RESAC* immer das erste Edukt durch das Produkt ersetzt. Das ist dann ein Unterschied, wenn nicht alle Moleküle gleichwahrscheinlich erster oder zweiter Reaktionspartner werden können. Die Wahl bestimmter räumlicher Topologien könnte das hervorrufen.

Obwohl – bezogen auf die natürliche Chemie – die Eduktersetzung vermutlich das realistischere Konzept darstellt, hat sich die freie Ersetzung (1) durchgesetzt bei Systemen, die nicht auf dem im vorletzten Abschnitt beschriebenen 'rewriting' beruhen. Allerdings wird diese Wahl kaum thematisiert, manchmal nicht einmal dokumentiert. Bekannte Künstliche Chemien mit Freiersetzung sind unter Anderem in (Hofbauer und Sigmund 1988; Adami und Brown 1994; Banzhaf 1994a; Fontana und Buss 1994; Banzhaf et al. 1999) zu finden. Die den Seceder-Effekt modellierende Künstliche Chemie von Dittrich et al. (2000) implementiert ebenfalls ausschließlich die freie Ersetzung.

Beispiele für Modelle variabler Dichte sind (Suzuki et al. 1996; Ono und Ikegami 1999; Ono und Ikegami 2001; Madina et al. 2003).

Die in (Hofbauer und Sigmund 1988) dargestellten Evolutionsexperimente im Flussreaktor (siehe Kap. 5) implementieren allesamt die freie Ersetzung. Die Autoren betonen dabei ihren Ansatz, nur die mathematischen – und nicht die chemischen – Aspekte studieren zu wollen. Diese Evolutionsexperimente gehen unter Anderem auf Eigen (1971), Eigen et al. (1981) und Küppers (1983) zurück. Bei diesen Experimenten tritt die Analogie zur Biologie in den Vordergrund – so ist die freie Ersetzung als durchaus realistisch einzuschätzen, beispielsweise wenn es um die Replikation von Polynukleotiden oder den Mechanismus der Zellteilungen geht. Die Replikation zerstört im Allgemeinen ihre Vorlage nicht.

Im anwendungsorientierten System *RESAC* ist die Realisierung beider Modelle möglich, keines widerspricht der zugrunde liegenden Philosophie. Es ist dabei a priori nicht klar, welche Vor- und Nachteile beide Strategien mit sich bringen. Das macht die grundlagenorientierte Analyse der Ersetzungsmethodik unvermeidlich.

[5] Im Folgenden wird der englische Begriff verwendet, um Missverständnisse zu vermeiden.

4.4 Zusammenfassung

Der Vergleich der beiden in *RESAC* angebotenen Ersetzungsstrategien wird anhand eines einfachen Basisexperiments durchgeführt, das drei Molekülsorten vorsieht. Es gibt nur einen einzigen Reaktionstyp: Ein Molekül des Typs s_o wird durch Kollision eines s_1- und s_2-Moleküls produziert. Diese einfache Wechselwirkung zwischen den Molekülen lässt eine Untersuchung der Ersetzungsmethodik offener oder geschlossener Reaktoren mit zum Teil analytischen Mitteln zu.

Die Merkmale lassen sie wie folgt zusammenfassen:

1. Frei- und Eduktersetzung unterscheiden sich in ihrem Wirkradius: Letztere wirkt lokal, während durch die freie Ersetzung auch an der jeweiligen Reaktion unbeteiligte Moleküle beeinflusst werden, also eine globale Wirkung entfacht wird.

2. Aus (1) entsteht für Eduktersetzung und geschlossene Reaktoren die Möglichkeit der Nischenbildung, die durch den Schutz passiver (nicht-reaktiver) Moleküle induziert ist. Solche nehmen nicht am Wettbewerb teil, solange keine Moleküle gebildet werden, mit denen sie reagieren können. Die durch Banzhaf (1995) dargestellten Möglichkeiten,[6] in konkurrenzbetonten Systemen zu überleben, erweitern sich für die Eduktersetzung um die Möglichkeit der hier beschriebenen „dynamischen Isolation“.

3. Die Konzentrationsentwicklungen in einem geschlossenen Reaktor mit Eduktersetzung sind stark abhängig von seiner Initialbefüllung, sie sind stark Anfangswert-sensitiv. Die Freiersetzung hingegen weist nur eine geringe Sensitivität auf, die Wahl der Startkonzentrationen der beteiligten Moleküle spielt kaum ein Rolle.

4. Nischenbildung ist auch in anwendungsorientierten Systemen wichtig, wenn Lösungen des zu bearbeitenen Problems solche Nischen besiedeln können. Das ist der Fall, wenn für Lösungskandiaten gilt, dass sie an keiner produktiven Kollision mehr teilnehmen.

5. Ein Öffnen der Reaktoren für einen festgelegten Zufluss aus Molekülen der Typen s_1 und s_2 wirkt der Nischenbildung und Anfangswert-Sensitivität entgegen.

6. Prinzipiell werden durch (5) häufig wünschenswerte Eigenschaften beschrieben, gerade bei der Eduktersetzung. Bei der Wahl der Zuflussrate ist jedoch die richtige Dosierung entscheidend. Eine zu hohe Zuflussrate bewirkt eine Verminderung der s_o-Konzentration.

7. Unter Zufluss ist die Eduktersetzung effizienter als die freie.

8. Bei offenen Reaktoren nimmt der Einfluss der Zeit auf die Konzentrationsverläufe exponentiell ab.

9. Zwei Fehlertypen wurden im Vergleich Frei-/ Eduktersetzung identifiziert. Strukturfehler und Konvergenzfehler sind abhängig von der Initialbefüllung.

[6] Die dort aufgezählten Möglichkeiten sind Selbstreplikation, Replikation durch die Hilfe anderer Moleküle und die Ausbildung von unterstützenden Reaktionszyklen wie die in Kap. 5 angesprochenen Hyperzyklen.

Kapitel 5

Katalytische Netzwerke in *RESAC*– *RESAC* als Evolutionsreaktor

Seit 1944 der Bakteriologe Oswald T. Avery nachgewiesen hat, dass die Erbinformation in Form von Polynukleotiden vorliegt, ist unumstritten, dass die Replikation von Polynukleotiden der elementare Prozess beim Vererbungsmechanismus ist und damit einen entscheidenden Mechanismus in der Evolution darstellt.

In der Natur gibt es mehrere Möglichkeiten, Polynukleotide zu vervielfältigen. Jedem Verfahren kann eine Komplexitätsschwelle zugeordnet werden. Der bedeutsamste Schritt ist hier der Übergang von der enzymfreien Replikation zur Replikation mit Enzymen, mit dem eine immense Steigerung der Kopiergenauigkeit einhergeht. Diese erhöhte Kopiergenauigkeit ermöglicht eine entsprechende Steigerung der maximalen Länge eines Moleküls. Andererseits erfordern komplexe Replikationsmechanismen sehr lange Polynukleotide, was enzymfreie Replikation ausschließt. Solche langen Nukleotidketten können aber nur bei geringen Fehlerraten bei der Replikation überdauern, sie erfordern schon die Existenz eines genaueren Replikationsmechanismus (Eigen 1971; Eigen und Schuster 1979). Dieses Paradoxon („Informationskrise“ genannt) motiviert die Analyse präbiotischer Evolution (Hofbauer und Sigmund 1988).

Der Untersuchung katalytischer Netzwerke kommt daher aus evolutionstheoretischer Sicht eine ganz besondere Bedeutung zu. Die Durchführbarkeit von Evolutionsexperimenten in einem System zeigt einerseits das Potenzial des dynamischen Systems in Hinblick auf die Untersuchung biochemischer und biomathematischer Fragen sowie Fragen der Evolutionstheorie. Andererseits gibt sie Hinweise auf das Wesen der zugrunde liegenden dynamischen Prozesse.

In diesem Kapitel wird darauf eingegangen, wie Evolutionsexperimente im System *RESAC* durchgeführt werden können. Die Bildung katalytischer Netzwerke sowie die Einschränkungen in resolutionsbasierten Systemen werden thematisiert. Dabei werden verschiedene Arten der Reaktionskopplung unterschieden.

5.1 Evolutionsexperimente in *RESAC*

Es ist naheliegend, die Untersuchung der Evolution von selbstreproduzierenden Polynukleotiden durch Experimente in einem Reaktor durchzuführen. Auf die Frage, was denn in seinen Evolutionsreaktoren passiere, antwortete Manfred Eigen in einem Interview:

> *Es sind Maschinen, in denen Prozesse einer „künstlichen" Evolution unter kontrollierten Bedingungen ablaufen.*
>
> Eigen (2002)

Eigen hatte im Jahre 1979 zusammen mit Peter Schuster die zyklische Verknüpfung von Reaktionen als Erklärung für die Selbstorganisation von präbiotischen Systemen beschrieben (Eigen und Schuster 1979). Die so genannten Hyperzyklen konservieren durch Rückkopplung Informationen, integrieren sie. Diese aus der Beschäftigung mit Evolutionsreaktoren stammenden Erkenntnisse beschreiben möglicherweise eine Lösung der Informationskrise. Das Konzept solcher Reaktoren stammt von Eigen (1971). Eine erste diskrete Version findet sich bei Spiegelman (1971).

Evolutionsreaktoren sind typischerweise chemische Flussreaktoren mit einer Zufuhr energiereicher Nährstoffe und einer Abfuhr energiearmer „Abfallprodukte". Sie haben daher grundsätzlich ähnlichen Aufbau wie die Künstliche Chemie *RESAC* (Kapitel 3). In der mathematischen Modellierung (siehe beispielsweise Hofbauer und Sigmund (1988)) wird das freie Ersetzungsschema verwendet und vom gut gerührten Reaktor ausgegangen, der durch einen *Verdünnungsfluss* in seiner Größe konstant gehalten wird. In diesem Kapitel wird untersucht, in wie weit sich Evolutionsexperimente in resolutionsbasierten Systemen durchführen lassen. Ein Aspekt wird dabei auch der Einfluss der Wahl des Eduktersetzungsschemas (siehe Kapitel 4) sein.

Im Folgenden werden Systeme betrachtet, in denen sich die Konzentrationen der Moleküle nur aufgrund von Wechselwirkungen mit anderen im Reaktor befindlichen Molekülen ändert. Die Ratengleichung lässt sich dann für Molekülsorten $M_1,\ldots,M_n$, die in relativen Konzentrationen $x_1,\ldots,x_n,\ 0\leq x_i\leq 1$, im Reaktor vertreten sind, aufschreiben (Hofbauer und Sigmund 1988):

$$\dot{x}_i = G_i x_i - \Phi x_i = x_i(G_i - \Phi),\ \text{für } i = 1,\ldots,n. \tag{5.1}$$

Dabei bezeichnet G_i die Wachstumsrate des Moleküls M_i und Φ den Abfluss aus dem Reaktor. Der Abflussterm wird so gewählt, dass die Gesamtkonzentration der Moleküle konstant bleibt. Zunahme (Wachstum) und Abnahme (Abfluss) einer Molekülsorte geschehen proportional zu deren Konzentration. Je nach Wesen der Wachstumsrate unterscheidet man verschiedene Fälle.

5.1.1 Arten der Kopplung

Ein besonders wichtiger Fall ist (1) die konstante Kopplung, bei der die Wachstumsraten G_i konstant sind (Hofbauer und Sigmund (1988), Gleichung 2.3 mit $a_i = 1, b_i = k_i + 1$):

$$\dot{x}_i = k_i x_i - \Phi x_i = x_i(k_i - \Phi),\ \text{für } i = 1,\ldots,n \text{ und } k_i > 0. \tag{5.2}$$

M_i vermehrt sich also proportional zur eigenen Konzentration x_i. Die Zahl der Moleküle M_i reduziert sich jedoch um den Abfluss, der ebenso proportional zu x_i ist.

Weitere Kopplungsarten, die unter der Bezeichnung *katalytische Netzwerke* zusammengefasst werden, sind: (2) Autokatalytische Kopplung, (3) Hyperzyklische Kopplung, (4) Katalytische Ketten.

Die Fälle (1) – (4) werden nun detailliert behandelt. Erwähnt sei noch der Fall, der allgemeinen linearen Kopplung, die einerseits durch konstanten Zuwachs gekennzeichnet ist, andererseits durch Linearkombination der Konzentrationen der anderen Moleküle erfolgt. Die Wachstumsrate G_i ist nach Hofbauer und Sigmund (1988):

$$\begin{aligned} G_i &= k_i x_i + (a_{i1}x_1)x_i + \ldots + (a_{in}x_n)x_i - \Phi x_i \\ &= x_i(k_i + \sum_{j=1}^{n} a_{ij}x_j - \Phi), \text{ für } i = 1,\ldots,n, \text{ und } k_i, a_{ij} \geq 0. \end{aligned}$$

5.2 Konstante Wachstumsraten

Bei der Replikation mit konstanten Wachstumsraten vermehrt sich jedes Molekül mit konstanter Geschwindigkeit, die nicht von den Konzentrationen anderer Moleküle abhängen, sondern nur durch die Konstanten k_i in Gleichung 5.2 gegeben ist. Dieser Fall entspricht so der enzymfreien Selbstreplikation.

Schreibt man Gleichung 5.2 in der vereinfachten chemischen Notation (wie sie schon in Kapitel 3 verwendet wurde), so ergibt sich für die freie Ersetzung und Molekülsorte M_1 mit zugeordneter Wachstumskonstante k_1 das folgende Schema (der Abfluss wird hier nur indirekt deutlich):

$$M_1 \longrightarrow M_1 + \overbrace{M_1 + \ldots + M_1}^{(k_1)-mal} \tag{5.3}$$

Dynamisches Verhalten Das dynamische Verhalten wird bestimmt durch die Molekülsorte mit der höchsten Wachstumsrate. Diese setzt sich auf Kosten der anderen durch. Alle molekularen Spezies mit niedrigerer Wachstumsrate werden komplett aus dem Reaktor verdrängt.

Vergleich Eduktersetzung Das Schema der Eduktersetzung lässt sich in Systemen mit konstanter Kopplung nicht anwenden, wenn – wie in *RESAC*– ein Reaktor mit konstanter Dichte modelliert wird. Die Zahl der Produkte kann dann die der Edukte nicht übersteigen. Damit gilt k_i=0 für alle Spezies M_i.

Vergleich Logiksystem Konstante Wachstumsraten entsprechen aus Sicht der Logik der spontanen Mehrfachkopie einzelner logischer Aussagen (Sätze). Dies hat keine substanzielle Änderung der Wissensbasis zur Folge und keinen Einfluss auf die Ableitbarkeit von Theoremen. Wenngleich diese Kopplung also aus dieser Perspektive sinnlos ist, ist sie jedoch nicht wirkungslos. Durch die Änderung der Konzentrationen im Reaktor wirkt sie sich auf die Dynamik aus.

5.3 Autokatalytische Selbstreplikation

Die autokatalytische Kopplung drückt sich durch folgende Gleichungen aus (Hofbauer und Sigmund (1988), Bisswanger (2000) für $k = 1$ und uniformer Reaktionsgeschwindigkeit, Gleichung 2.3 mit $a_i = 2, b_i = k_i + 2$):

$$\dot{x}_i = (k_i x_i)x_i - \Phi x_i = x_i(k_i x_i - \Phi), \text{ für } i = 1,\ldots,n \text{ und } k_i > 0, \tag{5.4}$$

$$\Phi = \sum_{i=1}^{n} (k_i x_i)x_i \tag{5.5}$$

Der Abfluss besteht aus der Summe aller hinzugewonnenen Moleküle. M_i wirkt so autokatalytisch auf die eigene Wachstumsrate. Dämpfung durch Konkurrenz um eine Replikase ist ein Beispiel. Die Gleichung entspricht bei freier Ersetzung dem folgenden Schema (hier für Sorte M_1):

$$M_1 + M_1 \longrightarrow M_1 + M_1 + \overbrace{M_1 + \ldots + M_1}^{(k_1)-mal} \tag{5.6}$$

Dynamisches Verhalten Hofbauer und Sigmund (1988) legen dar, dass das dynamische Verhalten durch die Anfangswerte bestimmt wird. Die Molekülsorte M_i mit größtem Produkt $k_i x_i$ wird sich zu Lasten der übrigen durchsetzen.

Vergleich Eduktersetzung Wie bei der konstanten Kopplung lässt sich hier das Schema der Eduktersetzung nicht anwenden, wenn ein Reaktor mit konstanter Dichte modelliert wird. Die Zahl der Produkte kann in einem solchen Fall die der Edukte nicht übersteigen. Damit gilt k_i=0 für alle Spezies M_i.

Vergleich Logiksystem Die autokatalytische Selbstreplikation entspricht einer (Mehrfach-) Kopie bei Kollision gleicher Edukte. Aus der Sicht eines auf Logik basierenden automatischen Beweisers ist das nicht sinnvoll, denn es erfolgt kein Zugewinn an Information. In einem Reaktor endlicher Größe hat die Selbstreplikation zwei Auswirkungen: Einerseits führt die Konzentrationserhöhung zu einem verbreiterten Angebot dieser Molekülsorte mit der Konsequenz, häufiger in nachfolgenden, möglicherweise nutzbringenden Reaktionen involviert zu sein. Andererseits werden andere Moleküle verdrängt und die von ihnen getragene Information vernichtet. Es wird sich zudem herausstellen, dass Moleküle, die in der besprochenen Weise verkoppelt sind, starken strukturellen Einschränkungen unterliegen.

Durch diese autokatalytische Kopplung wird eine ganz bestimmte Molekülstruktur induziert:

Satz 5.3.1 (1. Strukturtsatz)
Es gilt in der resolutionsbasierten Künstlichen Chemie RESAC *die folgende Implikation:*

Molekül M kann autokatalytisch selbstreplikativ wirken
$\Longrightarrow$ *M repräsentiert eine Klausel mit genau 2 Literalen entgegengesetzter Polarität.*

Beweis: Es bezeichne C die Klausel, die durch das Molekül M repräsentiert wird, und l_C die Zahl der Literale in C. Da M autokatalytisch wirken kann, folgt, dass M das Ergebnis einer produktiven Kollision zwischen M-Molekülen sein kann. Die Resolution zwischen zwei gleichen Klauseln C, deren Variablen umbenannt wurden (diese Umbenennung geht jeder Resolution voraus, siehe Bemerkung zu Def. 1.2.12), muss also erneut C ergeben können (bis auf Variablenumbenennung). Man betrachte die Länge der Ergebnisklausel, die die Summe der Eduktlängen ist, reduziert um die beiden Literale, die geschnitten werden. Da diese Klausel wieder C beschreiben muss, gilt:

$$l_C + l_C - 2 = l_C. \tag{5.7}$$

Es folgt direkt $l_C = 2$. Die Konstante 2 in Gleichung 5.7 folgt aus der Auflösung der im Falle der binären Resolution beiden unifizierten Literale.

Die entgegengesetzte Polarität der beiden Literale folgt direkt aus der vorausgesetzten Eigenschaft, autokatalytisch wirken zu können. Andernfalls wäre keine Resolution zwischen beiden Literalen der Klausel C anwendbar. □

Beispiel 5.3.1 (Autokatalyse in *RESAC*)
Anhand dreier Experimente wird gezeigt, dass *RESAC* die theoretische Dynamik bestätigt. Die folgenden vier Klauseln seien durch die Moleküle $M_1, \ldots, M_4$ repräsentiert:

$$M_1 = P(a)\overline{P(a)} \qquad M_2 = P(b)\overline{P(b)} \qquad M_3 = P(c)\overline{P(c)} \qquad M_4 = P(d)\overline{P(d)} \tag{5.8}$$

Die Abbildungen 5.1, 5.2 und 5.3 zeigen den Wettbewerb der Molekülsorten in einem *RESAC*-Experiment mit freier Ersetzung. Bis auf die Molekülsorte mit der höchsten Anfangskonzentration sterben alle Moleküle aus.

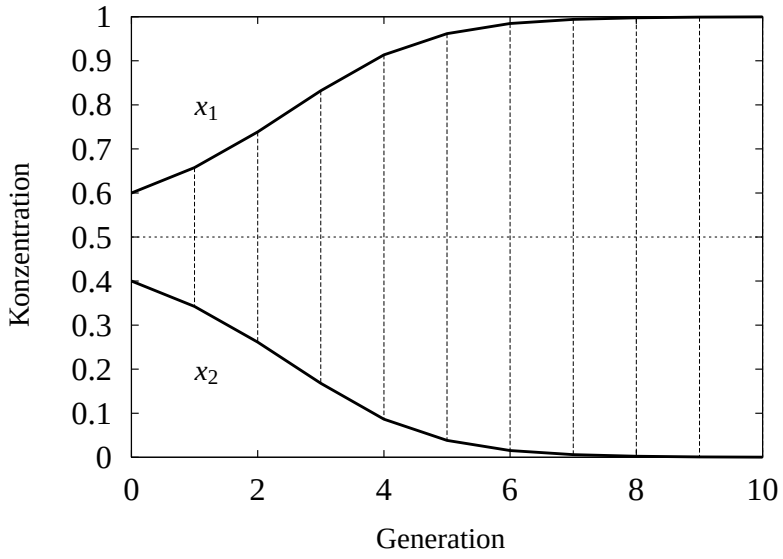


Abbildung 5.1: *RESAC*-Experiment mit den beiden in (5.8) definierten autokatalytischen Molekülen M_1 und M_2. Die Anfangskonzentrationen sind $x_1(0) = 60\%$ und $x_2(0) = 40\%$. Die Größe des Reaktors beträgt 100000 Moleküle, Ersetzungsschema ist die freie Ersetzung.

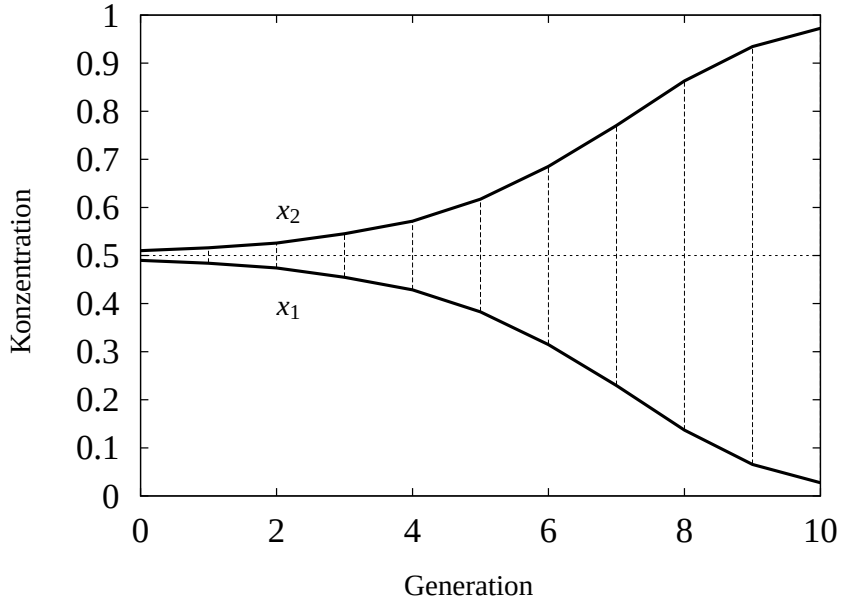


Abbildung 5.2: Experiment mit den in (5.8) definierten beiden autokatalytischen Molekülen M_1 und M_2 in *RESAC*. M_1 liegt anfangs in Konzentration $x_1(0) = 49\%$ vor, M_2 in Konzentration $x_2(0) = 51\%$. Die Größe des Reaktors beträgt 100000 Moleküle, freie Ersetzung.

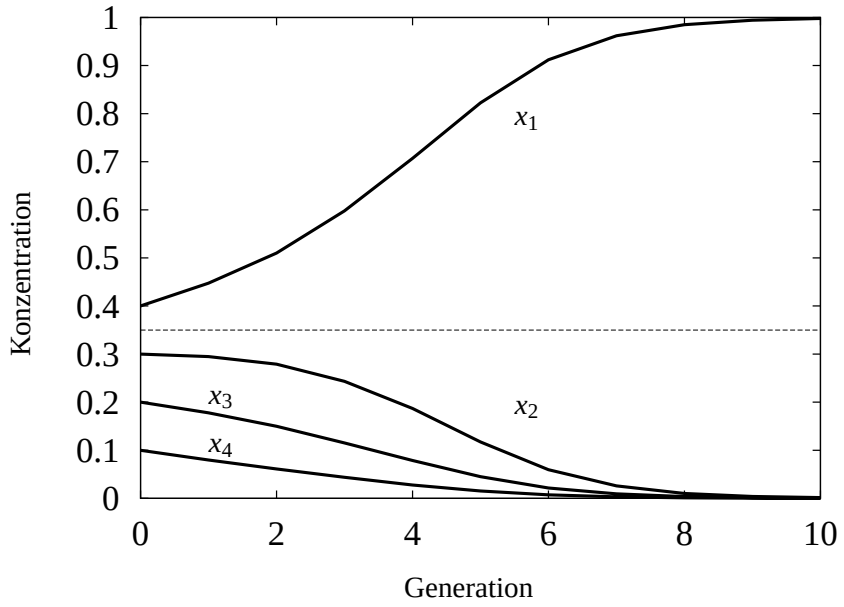


Abbildung 5.3: Experiment mit den in (5.8) definierten autokatalytischen Molekülen in *RESAC*. Der Konzentrationsvektor $(x_1(0), x_2(0), x_3(0), x_4(0))$ beträgt $(0.4, 0.3, 0.2, 0.1)$. Die Größe des Reaktors beträgt 200000 Moleküle, Ersetzungsschema ist die freie Ersetzung.

5.4 Hyperzyklus

Den beiden elementaren Kopplungen aus den vorangegangenen Kapiteln ist gemein, dass sich in den Fällen, in denen es eine Sorte gibt, die eine höhere Konzentration als alle übrigen Sorten aufweist, sich diese Sorte durchsetzen und alle anderen verdrängen wird. Damit kann die angesprochene Informationskrise nicht gelöst werden. Eigen und Schuster (1979) schlugen eine Kopplung vor, in der Information konserviert wird. Der Sieger des Wettbewerbs vernichtet dabei nicht alle übrige Information. Erreicht wird dies durch einen Rückkopplungskreis, *Hyperzyklus* genannt. Eine Kooperation der Moleküle entsteht durch einen geschlossenen Kreis katalytischer Wirkungen.[1] Die Ratengleichung für den Hyperzyklus lautet bei freier Ersetzung (nach Hofbauer und Sigmund (1988)):

$$\begin{aligned} \dot{x}_i &= (k_i x_{i-1})x_i - \Phi x_i = x_i(k_i x_{i-1} - \Phi), \text{ für } i = 1,\ldots,n \text{ und } k_i > 0, \\ &\Phi = \textstyle\sum_{i=1}^{n} (k_i x_{i-1})x_i. \end{aligned} \tag{5.9}$$

Die Selbstreplikation des Moleküls M_i wird durch das Molekül M_{i-1} katalysiert, hängt daher proportional von seiner Konzentration x_{i-1} ab. Wichtig ist, dass die Indizes zyklisch gezählt werden, d. h. 0 wird mit n identifiziert. So bildet M_n das Vorläufermolekül für M_1. Das Schema lautet allgemein:

$$M_i + M_{i-1} \longrightarrow M_i$$

Dynamisches Verhalten Wie schon beschrieben stellt sich durch die Rückkopplung eine Kooperation der Moleküle ein. Keines stirbt aus.

Vergleich Eduktersetzung Das Schema der Eduktersetzung lässt sich im Gegensatz zur konstanten oder autokatalytischen Kopplung bei der hyperzyklischen anwenden. Der Reaktionszuwachs M_i ersetzt dann ein Edukt: Das Reaktionsschema lautet in vereinfachter chemischer Notation:

$$x_i + x_{i-1} = \begin{cases} x_i + x_i & \text{(1) Zuwachs ersetzt Molekül der Sorte } M_{i-1}, \\ x_i + x_{i-1} & \text{(2) Zuwachs ersetzt Molekül der Sorte } M_i. \end{cases}$$

Beide Fälle treten gleichwahrscheinlich auf. Nur im Fall (1) findet eine Replikation statt. Im Vergleich zur Ratengleichung der freien Ersetzung hat man es daher mit einer Geschwindigkeitsänderung (Halbierung) zu tun.

Vergleich Logiksystem Hier ist Ähnliches wie bei den vorher besprochenen Kopplungen festzustellen. Für einen automatischen Theorembeweiser ist es im Allgemeinen nicht sinnvoll, schon vorhandene, das heißt schon hergeleitete Information, zu replizieren, da kein Informationszugewinn erfolgt. Es wird sich außerdem im Folgenden herausstellen, dass Moleküle, die logische Information tragen und in resolutionsbasierten Systemen wie *RESAC* rückgekoppelt sind, strukturelle Einschränkungen aufweisen.

[1] Der Beweis findet sich in (Schuster et al. 1979).

Durch die hyperzyklische Kopplung wird eine Molekülstruktur induziert, die zur Folge hat, dass in *RESAC* niemals Hyperzyklen entstehen wie sie alleinig durch Gleichung 5.9 beschrieben werden können. Bevor dies im nächsten Kapitel bewiesen wird, erfolgt noch eine Aussage zur grundsätzlichen Struktur von Molekülen, die in *RESAC* an Hyperzyklen beteiligt sind.

Satz 5.4.1 (2. Struktursatz)
Gäbe es in dem resolutionsbasierten System RESAC *einen Hyperzyklus, so besäßen die beteiligten Moleküle genau zwei Literale von entgegengesetzter Polarität.*

Beweis: Man betrachte einen (nach Voraussetzung existierenden) Hyperzyklus der Länge n in *RESAC* über Moleküle $M_1,\ldots,M_n$, die Klauseln $C_1,\ldots,C_n$ repräsentieren. Der katalytische Kopplungskreis und die Gleichung für die Klausellängen l_{c_i} sehen dann folgendermaßen aus:

$$\left.\begin{array}{r} M_1+M_2=M_2 \\ M_2+M_3=M_3 \\ \vdots \\ M_{i-1}+M_i=M_i \\ \vdots \\ M_{n-1}+M_n=M_n \\ M_n+M_1=M_1 \end{array}\right\} \; l_{C_{i-1}}+l_{C_i}-2=l_{C_i} \Leftrightarrow l_{C_i}=2 \;\text{ für }\; i=1,\ldots,n$$

Es folgt $l_{C_i}=2$ für alle Klauseln C_i. Die Konstante 2 in der Längengleichung folgt aus der Auflösung der – im Falle der binären Resolution – beiden unifizierten Literale.

Wegen dieser Längenbeschränkung kommen nur bestimmte Polaritätsverteilungen in Frage. Es bezeichne '+' ein positives Literal, '-' ein negiertes Literal, '++' zwei positive Literale usw. Es sei M_{Erg} das Ergebnis der Resolution zwischen M_{i-1} und M_i (kommutative Möglichkeiten sind nur einmal aufgezählt):

Polaritäten des Literalpaares aus Klausel	M_{i-1}	M_i	M_{Erg}	$M_{Erg}=M_i$
(a)	$++$	$--$	$+-$	*nein*
(b)	$++$	$+-$	$++$	*nein*
(c)	$++$	$++$	—	*nein*
(d)	$+-$	$--$	$--$	*ja*
(e)	$+-$	$+-$	$+-$	*ja*
(f)	$+-$	$++$	$++$	*ja*
(g)	$--$	$--$	—	*nein*
(h)	$--$	$+-$	$--$	*nein*
(i)	$--$	$++$	$+-$	*nein*

Nur die Polaritätenkombinationen (d)-(f) erlauben eine korrekte Replikation der M_i. Weiterhin scheiden die Kombinationen (d) und (f) aus, da Klauseln mit Polaritätskombinationen $--$ und $++$ niemals als Katalysator fungieren könnten, sie verlassen den Kombinationskomplex (d)-(f). Also erlaubt nur Schema (e) die hyperzyklische Kopplung. Es folgt für alle Klauseln C_i, dass sie aus zwei Literalen entgegengesetzter Polarität bestehen. □

5.4.1 Satz über die Nichtexistenz von Hyperzyklen in *RESAC*

Der folgende Satz wird weitere strukturelle Attribute hyperzyklisch gekoppelter Klauseln zeigen, wenn die Reaktionsvorschrift durch die Anwendung der Resolution gegeben ist: Niemals treten in *RESAC* Hyperzyklen auf; eine autokatalytische Kopplung der Elemente lässt sich nicht verhindern.

Die folgenden Definitionen und das Lemma leiten den Satz ein. Sie basieren auf dem Begriff der Substitution und Umbenennung (siehe Def. 1.2.10).

Definition 5.4.1 (Wertebereich einer Substitution)
Der *Wertebereich WB(σ)* einer Substitution σ bezeichnet die Menge der Substitutionsterme, die durch die Anwendung von σ den Variablen zugeordnet werden.

Definition 5.4.2 (Umkehr-Umbenennung, Identitäts-Substitution ε)
Die *Identitäts-Substitution* wird durch ε notiert. Für alle Terme t des betrachteten Diskurses gilt: $t\varepsilon = t$. Da in dieser Arbeit nur die Wirkung von ε interessant ist, nicht jedoch die Struktur, wird im Folgenden (vereinfachend) vereinbart: Gilt $t\alpha = t$ für alle Terme t des Diskurses und Substitution α, so wird notiert: α=ε.

Eine Umbenennung σ^{-1} heißt genau dann *Umkehr-Umbenennung*, wenn durch Anwendung auf eine Umbenennung σ die Identitäts-Substitution beschrieben wird:

$$\sigma\sigma^{-1} = \varepsilon.$$

Lemma 5.4.1 (Umkehrung einer Umbenennung)
Jede Umbenennung ist umkehrbar.

Beweis: Eine Umbennung σ von n Variablen wird beschrieben durch $\{\text{Quelle}_i/\text{Ziel}_i\}_{i=1,\ldots,n}$. Nach Def. 1.2.10 gilt:

$$\forall i,j \in \{1,\ldots,n\}: \quad 1.\quad \text{Quelle}_i = \text{Quelle}_j \;\Rightarrow\; i = j$$
$$2.\quad \text{Ziel}_i = \text{Ziel}_j \;\Rightarrow\; i = j.$$

Die Umbenennung $\sigma^{-1} = \{\text{Ziel}_i/\text{Quelle}_i\}_{i=1,\ldots,n}$ ist dann die Umkehr-Umbenennung von σ. Es gilt:

$$\sigma\sigma^{-1} = \{\text{Quelle}_i/\text{Quelle}_i\}_{i=1,\ldots,n} = \varepsilon.$$

□

Satz 5.4.2 (3. Struktursatz)
Es gibt keine Hyperzyklen in dem resolutionsbasierten System RESAC. *In diesem System induziert die katalytische Replikation unter einem Katalysator K die autokatalytische Selbstreplikation von K.*

Die grobe Idee ist, dass die Literale einer Katalysator-Klausel M_{i-1} strukturell einander ähnlich sein müssen, da eines davon genau *das* Literal der replizierten Klausel M_i ersetzt, welches zuvor mit dem anderen, also dem Partnerliteral aus M_{i-1} unifizierte. Die Ähnlichkeit der Struktur ermöglicht eine autokatalytische Reaktion. Es folgt der formale Beweis:

Beweis:

Annahme: Es gibt Hyperzyklen in *RESAC*.

Ausgangspunkt des Beweises ist die Gleichung der hyperzyklischen Kopplung der n Moleküle $M_1, \ldots, M_n$:

$$M_i + M_{i-1} = M_i, \text{ für } i = 1, \ldots, n. \tag{5.10}$$

Die durch diese Moleküle repräsentierten Klauseln haben nach dem 2. Struktursatz 5.4.1 genau zwei Literale von entgegengesetzter Polarität. Die Literale seien o. B. d. A. wie folgt bezeichnet:

$$\overbrace{l_{i,1}\ \overline{l_{i,2}}}^{M_i} \ + \ \overbrace{l_{i-1,1}\ \overline{l_{i-1,2}}}^{M_{i-1}} \ = \ \overbrace{l_{i,1}\ \overline{l_{i,2}}}^{M_i} \quad \text{für } i = 1, \ldots, n. \tag{5.11}$$

Der Übersichtlichkeit wegen wird dabei $\overline{l_x}$ notiert, wenn zum Ausdruck gebracht werden soll, dass $\overline{l_x}$ umgekehrte Polarität hat zu l_x. Der Betragsoperator invertiert die vorliegende Polarität: $l_x = |\overline{l_x}|$. Eine Zusammenfassung der weiteren im Beweis verwendeten Symbole ist Tabelle 5.1 zu entnehmen.

O. B. d. A. komme die hyperzyklische Kopplung gemäß Gleichung 5.10 durch Resolution der Literale $l_{i,1}$ und $\overline{l_{i-1,2}}$ zu Stande. Per definitionem gilt mit Substitution σ und Umbenennungen η, γ:

I	$l_{i,1}\sigma = \lvert\overline{l_{i-1,2}}\rvert\eta\sigma$	Dies folgt aus Unifizierbarkeit von $l_{i,1}$ und $\overline{l_{i-1,2}}$.
II	$l_{i-1,1}\eta\sigma\gamma = l_{i,1}$	Literal $l_{i-1,1}$ muss nach der Resolution (der die Anwendung von $\eta\sigma$ entstammt) das durch die Resolution „aufgebrauchte“ Literal $l_{i,1}$ ersetzen, d. h. bis auf Variablenumbenennung gleich sein, die γ durchführt. Literal $\overline{l_{i,2}}$ kann das wegen umgekehrter Polarität nicht.
I+II=III	$l_{i-1,1}\eta\sigma\gamma\sigma = \lvert\overline{l_{i-1,2}}\rvert\eta\sigma$	Hergeleiteter Zusammenhang zwischen $l_{i-1,1}$ und $\overline{l_{i-1,2}}$.

Zu zeigen ist die Unifizierbarkeit von $l_{i-1,1}$ und $\overline{l_{i-1,2}}$, denn daraus folgt, dass M_{i-1} mit sich selbst reagieren kann, nämlich durch eine Resolution zwischen $l_{i-1,1}$ und $\overline{l_{i-1,2}}$.

Ausgehend von (III)

$$l_{i-1,1}\eta\sigma\gamma\sigma = |\overline{l_{i-1,2}}|\,\eta\sigma \tag{5.12}$$

wird die Identitäts-Substitution ε eingeführt, die neutral Variablen auf sich selbst abbildet:

$$l_{i-1,1}\eta\,\varepsilon\sigma\gamma\sigma = |\overline{l_{i-1,2}}|\,\eta\sigma.$$

ε wird nun durch eine spezielle Umbenennung δ mit den in Tabelle 5.2 vereinbarten Eigenschaften und der zugehörigen Umkehr-Umbenennung δ^{-1} äquivalent ersetzt, da $\delta\delta^{-1} = \varepsilon$:

$$l_{i-1,1}\eta\,(\delta\delta^{-1})\,\sigma\gamma\sigma = |\overline{l_{i-1,2}}|\,\eta\sigma.$$

Die Notation der Klammern dient dabei nur der Übersichtlichkeit und hat keinen Einfluss auf die Rechnung. Eine andere Lesart derselben Gleichung liefert äquivalent:

$$l_{i-1,1}\eta\delta\underbrace{(\delta^{-1}\sigma\gamma\sigma)}_{\tau_\delta} = |\overline{l_{i-1,2}}|\,\eta\underbrace{\ \sigma\ }_{\tau_\eta}. \tag{5.13}$$

Mit Substitutionen $\tau_\delta = \delta^{-1}\sigma\gamma\sigma$ und $\tau_\eta = \sigma$ sowie

$$\tau = \begin{cases} \tau_\delta & \text{für Variablen } v \in WB(\delta), \\ \tau_\eta & \text{sonst,} \end{cases} \tag{5.14}$$

deren Gültigkeit später bewiesen wird, gilt schließlich mit Gleichung 5.13:

$$(l_{i-1,1}\eta)\,\delta\tau = (|\overline{l_{i-1,2}}|\,\eta)\,\tau. \tag{5.15}$$

Nach der Definition der Resolution (Def. 1.2.12) sind $l_{i-1,1}\eta$ und $|\overline{l_{i-1,2}}|\,\eta$ resolvierbar mit Umbenennung δ und Unifikator τ. Da η eine Umbenennung ist und auf die Literale derselben Klausel Anwendung findet, sind auch die Literale $l_{i-1,1}$ und $\overline{l_{i-1,2}}$ resolvierbar. Daraus folgt die autokatalytische Kopplung des zugehörigen Moleküls M_{i-1}, und die Ratengleichung kann nicht durch Gleichung 5.9 ausgedrückt werden. Das ist ein Widerspruch zur Annahme. Da jedes Molekül des Zyklus auch als Katalysator fungiert, gilt die durchgeführte Betrachtung für alle Moleküle M_i.

Es bleibt zu zeigen, dass

(1) eine Umbenennung δ mit den geforderten Eigenschaften (siehe Tabelle 5.2) konstruiert werden kann.

Konstruktion δ / δ^{-1}:

Es seien $v_1, \ldots, v_k$ alle in $l_{i-1,1}\eta$ vorkommenden Variablen. Man wähle k neue Variablennamen $w_1, \ldots, w_k$, die niemals vergeben wurden und niemals erneut vergeben werden. Dann werden δ und δ^{-1} wie folgt konstruiert:

$$\delta = \bigcup_{j=1}^{k} \{v_j \,/\, w_j\} \qquad\qquad \delta^{-1} = \bigcup_{j=1}^{k} \{w_j \,/\, v_j\}.$$

Die Eigenschaften 1-3 aus Tab. 5.2 sind durch Konstruktion erfüllt. Eigenschaft 4 ist durch Lemma 5.4.1 gezeigt. Die letzte Eigenschaft folgt nach Definition der Umkehr-Umbenennung.

(2) τ eine gültige Substitution darstellt und die Substitution $\delta^{-1}\sigma\gamma\sigma$ genau auf die Variablen aus $l_{i-1,1}\eta\delta$ anwendet und σ auf die aus $|\overline{l_{i-1,2}}|\,\eta$.

Korrektheit der Substitution τ

Da δ^{-1}, σ und γ Substitutionen sind, ist auch τ eine endliche Menge von Zuordnungen der Form Variable/Term, in denen innerhalb einem einer Variablen zugeordneten Term diese Variable nicht wieder auftritt. Es bleibt nach Def. 1.2.10 zu zeigen, dass jeder Variable höchstens ein Term zugeordnet wird. Das aber ist durch die Eindeutigkeit der Fallunterscheidung gegeben (entweder eine Variable ist aus $WB(\delta)$ oder nicht) zusammen mit der Tatsache, dass das schon für δ^{-1}, σ und γ galt.

Da nach Tab. 5.2 Punkt 3 alle Variablen des Wertebereichs von δ Unikate sind, also nirgends sonst im System auftauchen, und zweitens nach Eigenschaft 2 genau die Variablen aus $l_{i-1,1}\eta$ durch δ umbenannt werden, wird nach 5.14 korrekt zugeordnet.

□

η	Standardumbenennung für anschließende Resolution.
σ	Unifikator für die Resolution.
γ	Umbenennung zur formalen Gleichsetzung logisch äquivalenter Klauseln (Normierung).
δ	Umbenennung mit den in Tab. 5.2 aufgelisteten Eigenschaften.
δ^{-1}	Umkehr-Umbenennung mit $\delta\delta^{-1} = \varepsilon$.
ε	Identiäts-Substitution (siehe Def. 5.4.2).
τ_δ	Substitution, angewendet nach Umbenennung δ.
τ_η	Substitution, angewendet nach Umbenennung η.
WB(θ)	Funktion, die einer Substitution θ ihren Wertebereich gemäß Def. 5.4.1 zuordnet.

Tabelle 5.1: Übersicht über die verwendeten Symbole

1. δ ist eine Umbenennung.
2. Genau die Variablen aus $l_{i-1,1}\eta$ werden umbenannt.
3. Die Variablen aus dem $WB(\delta)$ sind Unikate, d. h. ohne vorherige Anwendung von δ tauchen sie niemals im System auf.
4. δ^{-1} existiert.
5. $\delta\delta^{-1} = \varepsilon$.

Tabelle 5.2: Die Eigenschaften der Substitution δ.

Beispiel 5.4.1 (Verdeutlichung der Substitutionen im Satz 5.4.2)
Es ist

$$l_{i,1} = P(W, f(z(g)), h(X,a)) \qquad l_{i-1,1} = P(T, f(z(g)), h(Y,Z)) \qquad \overline{l_{i-1,2}} = \overline{P(f(a), X, h(Y,Z))},$$

dabei bezeichnen genau die Großbuchstaben die Variablen. Das Literal $\overline{l_{i,2}}$ ist bei dieser Betrachtung ohne Bedeutung. Bei Resolution zwischen $l_{i,1}$ und $\overline{l_{i-1,2}}$ ist

$$\begin{aligned}
\eta &= \{T/T', X/X', Y/Y', Z/Z'\},\\
\sigma &= \{W/f(a), X'/f(z(g)), X/Y', Z'/a\},\\
\gamma &= \{T'/W, Y'/X\},\\
\delta &= \{T'/T'', Y'/Y'', Z'/Z''\},\\
WB(\delta) &= \{T'', Y'', Z''\},\\
\delta^{-1} &= \{T''/T', Y''/Y', Z''/Z'\},\\
\tau_\delta &= \{T''/f(a), Y''/Y', Z''/a, W/f(a), X'/f(z(g)), X/Y', Z'/a, T'/f(a)\},\\
\tau_\eta &= \{W/f(a), X/Y', X'/f(z(g)), Z'/a\},\\
\tau &= \{T''/f(a), W/f(a), X/Y', X'/f(z(g)), Y''/Y', Z'/a, Z''/a\}.
\end{aligned}$$

5.5 Katalytische Ketten

Im Gegensatz zum Hyperzyklus liegt bei katalytischen Ketten kein geschlossener katalytischer Wirkungskreis vor. Molekülsorte M_1 katalysiert die Selbstreplikation von M_2, M_i die von M_{i+1} für $i = 1,\ldots,n-1$. Die katalytische Wirkung endet bei M_{n-1}. M_n katalysiert nun *nicht* die Replikation von M_1.

Wie Hyperzyklen sind auch katalytische Ketten nicht im resolutionsbasierten System *RESAC* konstruierbar. Beim Versuch, eine katalytische Kette aufzubauen, gehen stets weitere Kopplungen („parasitäre“ Kopplungen) einher. Der 2. Struktursatz 5.4.1 (Satz über die Struktur der an Hyperzyklen beteiligten Moleküle) sowie Satz 5.4.2 über die Nichtexistenz von Hyperzyklen lassen sich im Wesentlichen samt Beweis übertragen. Die Aussagen galten dort für alle Katalysatoren. Im Gegensatz zu den Hyperzyklen gelten sie bei katalytischen Ketten so nur noch für die Sorten $M_1,\ldots,M_{n-1}$, da M_n nicht mehr katalytisch auf M_1 wirkt.

5.6 Allgemeine Replikatorgleichung

Verallgemeinernd lassen sich Reaktionen eines Netzwerks von replizierenden Entitäten[2] durch die bekannte *Replikatorgleichung* (Taylor und Jonker 1978; Schuster und Sigmund 1983) beschreiben:

$$\begin{gathered}\dot{x}_i = x_i(f_i(x_1,\ldots,x_n) - \Phi(x_1,\ldots,x_n)), \text{ für } i = 1,\ldots,n. \\ \Phi(x_1,\ldots,x_n) = \textstyle\sum_{i=1}^{n} x_i f_i(x_1,\ldots,x_n).\end{gathered} \tag{5.16}$$

Die Funktion f_i modelliert den katalytischen Einfluss aller anderen Moleküle auf die Replikationsrate von x_i. Der Abflussterm $\Phi(x_1,\ldots,x_n)$ kann nach Hofbauer und Sigmund (1998) auch als durchschnittliche Fitness interpretiert werden.

Beispiel 5.6.1 (Katalytische Netzwerke)
Es seien 5 Molekülsorten gegeben:

$$M_1 = P(X)\overline{P(X)} \quad M_2 = P(a)\overline{P(a)} \quad M_3 = P(X)\overline{P(a)} \quad M_4 = P(Y)\overline{P(X)} \quad M_5 = P(a)\overline{P(X)}.$$

Das zugehörige Reaktionsnetzwerk[3] enthält neben vielen anderen Reaktionen[4] einen Rückkopplungskreis (zum Beispiel über M_5 und M_2), eine Kette von Katalysatoren ($M_1 \longrightarrow M_2 \longrightarrow M_3 \longrightarrow M_4 \longrightarrow M_5$) und autokatalytische Kopplungen. M_1 ist jedoch isoliert. Außer durch Autokatalyse ist keine Replikation von M_1 möglich. Es muss daher aussterben (siehe Abb. 5.4), während die anderen Sorten durch Rückkopplung konserviert werden. In einer reinen katalytischen Kette, würden alle Glieder bis auf das Endglied aussterben.

[2] Abseits der replizierenden Moleküle können durch die Gleichung 5.16 auch andere Systeme wie u. a. Öko-, Sozial- und Wirtschaftssysteme modelliert werden.

[3] Es ist die Umbenennung der Variablen zu beachten, so dass die Variablen der Edukt-Klauseln disjunkt sind (→ *Variablen standardisieren*).

[4] Beispielsweise reagiert Sorte M_2 mit Sorte M_4 zu M_3, oder auch M_5 mit M_2 zu M_2.

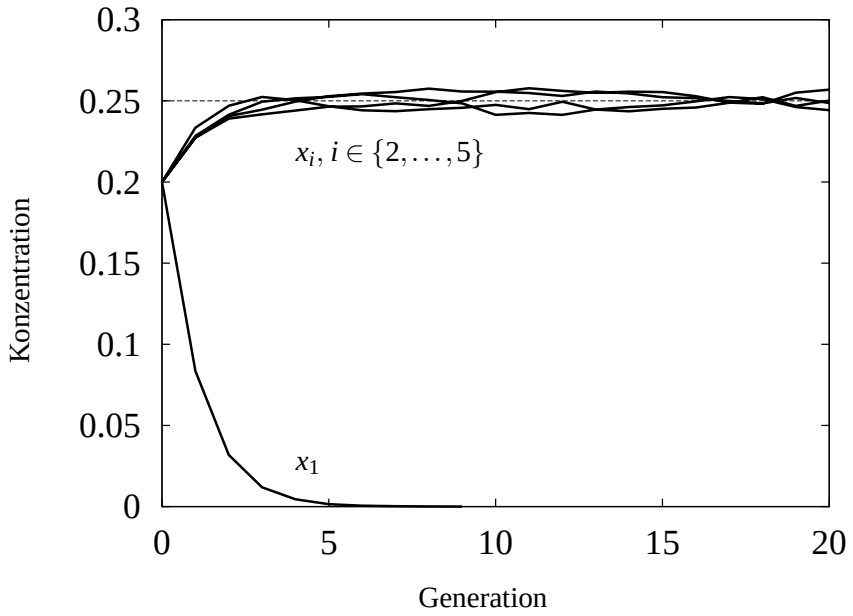


Abbildung 5.4: Katalytisches Netzwerk aus Beispiel 5.6.1 im *RESAC*-Experiment unter freier Ersetzung. Der Reaktor besteht aus 50000 Molekülen. Molekülsorte M_1 stirbt aus, die anderen Sorten konzentrieren sich zu gleichen Teilen.

5.7 Zusammenfassung

In diesem Kapitel wurde die Eignung der resolutionsbasierten Künstlichen Chemie *RESAC* im Hinblick auf die Durchführbarkeit von Evolutionsexperimenten untersucht. Hierbei geht es im Besonderen um den Aufbau katalytischer Netzwerke und die Koexistenz bzw. Integration von Information. Solche Experimente sind von großer Bedeutung für das Verständnis von biochemischen Evolutionsprozessen.

Eine erste Erkenntnis besteht darin, dass das Ersetzungsschema Einfluss hat auf die Bildungsmöglichkeiten solcher Netzwerke. Zumindest in dichtegleichen Systemen wie *RESAC* bedeutet die Eduktersetzung in diesem Sinne eine bedeutende Einschränkung.

Unter freier Ersetzung wären Autokatalyse, Hyperzyklen und katalytische Ketten auch in *RESAC* prinzipiell konstruierbar. Allerdings induziert das Kalkül der Prädikatenlogik 1. Ordnung mit seinem Klauselaufbau und der Resolution eine so stark eingeschränkte Molekülstruktur, dass von den genannten Mechanismen nur die Autokatalyse in reiner Form auftauchen kann. Diese Struktureinschränkung hat zwei Auswirkungen: Einerseits ist nur ein kleiner Ausschnitt der Logikrepräsentation nutzbar, andererseits weisen die nutzbaren Klauseln starke Ähnlichkeit untereinander auf (siehe Beispiel 5.6.1).

Der letzte Aspekt macht *RESAC*-Logiksysteme vergleichsweise unattraktiv für Evolutionsexperimente, da durch Seiteneffekte, „parasitäre“ Verkettungen und Zwangsverkopplungen ein wenig transparentes und darüber hinaus kaum gezielt zu konstruierendes Netzwerk entsteht.

Auf der anderen Seite sind Evolutionsexperimente im Logikreaktor im Allgemeinen wenig sinnvoll, da Replikation von Information beim automatischen Beweisen wenig nutzbringend ist. Hier geht es eher um den Verbrauch von Information zur Generierung neuer Schlussfolgerungen.

Zusammenfassend kann gesagt werden, dass *RESAC* weniger für Evolutionsexperimente oder Experimente zur Selbstorganisation geeignet ist. Ungeachtet dessen ist aber die Untersuchung von Replikationsprozessen lehrreich und enthüllt Eigenschaften resolutionsbasierter Systeme.

Teil III

Anwendungen

Kapitel 6

Automatisches Theorembeweisen mit *RESAC*

Das automatische Theorembeweisen, engl. automated theorem proving (ATP), ist eine der ältesten Disziplinen der Künstlichen Intelligenz. Möglich wurde eine mechanisierte Durchführung von Beweisen vor allem durch zwei Entwicklungen: die Kalkülisierung der Logik und die Realisierung schneller Computer. Die wohl wichtigsten Beiträge zur Bildung geeigneter Kalküle lieferte Gottlob Frege. In seinem Buch „Begriffsschrift“ (Frege 1882) entwickelte er die erste vollständige Formulierung dessen, was heute als Prädikatenlogik 1. Ordnung bezeichnet wird (Bläsius und Bürckert 1992). Sie liegt heute vielen Deduktionssystemen zugrunde.

Das System *RESAC* ist durch die prädikatenlogische Struktur seiner Moleküle und durch die Reaktionsvorschrift der Resolution prädestiniert für das automatische Beweisen. Obwohl *RESAC* nicht als Theorembeweiser konzipiert wurde, ist die automatische Deduktion durch die Einsetzung einer auf der Resolution basierenden Reaktion eine Anwendung dieser Künstlichen Chemie. Ein Vorteil liegt in der „massiven“ Parallelisierbarkeit des Inferenzprozesses. Eine Verteilung auf Milliarden Recheneinheiten ist möglich. Ein Beweis ist das Ergebnis eines emergenten Lösungsprozesses dieser parallel arbeitenden Einheiten. Solche Recheneinheiten können durchs Internet verbundene Computer sein. In der Fortführung der DNA-Experimente von Leonard Adleman (Adleman 1994) stellt das „Rechnen“ mit Molekülen eine andere Möglichkeit dar.

Dieses Kapitel führt eine Untersuchung des Suchraums durch, der durch das Theorembeweisen aufgespannt wird. Daran anschließend wird ein konventioneller Beweiser vorgestellt. Das darauf folgende Unterkapitel präsentiert die notwendigen Ergänzungen, die *RESAC* zum Beweiser machen. Es folgt ein Vergleich mit Prolog, der vielleicht bekanntesten logischen Programmiersprache. Abgeschlossen wird das Kapitel mit Beispielen, die das Theorembeweisen mit *RESAC* vorführen. Anhand dieser Beispiele wird auf interessante Teilaspekte eingegangen.

6.1 Suchraum

Automatisches Beweisen mit Hilfe der Resolutions-Inferenz ist aufwändig und führt zur kombinatorischen Explosion. Fragt man nach der Komplexität des Automatischen Beweisens, so be-

zieht man sich üblicherweise auf den eher intuitiven Begriff des Suchraums. Er charakterisiert den Raum aller möglichen Herleitungen, in dem nach einem Beweis gesucht werden muss. Eine Möglichkeit der Abschätzung der Größe des Suchraums stellt die Sortierung der gegebenen sowie der daraus herleitbaren neuen Klauseln mittels der Ordnungsstrategie Stufensättigung dar. Es handelt sich dabei um eine systematische Generierung aller Resolventen. Ordnet man auf einer Ebene – der nullten Ebene – alle durch das Problem gegebenen Klauseln (*Problemklauseln* genannt) an, auf der nächsten Ebene zusätzlich zu diesen Problemklauseln alle daraus ableitbaren und allgemein auf Ebene k alle Klauseln mit Herleitungslänge k an, so wird durch die Resolutionen der Suchraum vollständig aufgespannt. Durch das Konstruktionsverfahren wird klar, dass seine Größe mit der Zahl der Ebenen exponentiell wächst: Selbst wenn pro Klauselpaar nur eine Resolution anwendbar ist und sich n Klauseln auf der Ebene k befinden, so weist Ebene $k+1$ maximal $n+n+\binom{n}{2} = n+\left(\frac{n(n+1)}{2}\right)$ Klauseln auf. Der erste Summand zählt die aus der Vorgängerebene übernommenen Klauseln, der zweite die aus Resolution mit sich selbst entstehenden Klauseln.

Bestehen die Problemklauseln aus höchstens m Literalen, so kann nach Goubault-Larrecq und Mackie (1997) die Zahl der Klauseln auf Ebene k bis $2^k mn^{2^k}$ ansteigen, wobei die Klauseln dann aus bis zu $2^k m$ Literalen bestehen können. Auf der anderen Seite ist durch Haken (1985) und Nachfolgearbeiten bekannt, dass es Probleme gibt, die tatsächlich die Herleitung von exponentiell vielen Zwischenklauseln benötigen.

RESAC kann – wie andere Beweiser auch – aufgrund der begrenzten Reaktorgröße nicht den gesamten Suchraum absuchen. Es fehlt zudem ein Speicher, in dem alle bis zu einem bestimmten Zeitpunkt abgeleiteten Klauseln nachgehalten werden. Zwischenergebnisse werden durch Zufluss oder Reaktionsprodukte überschrieben, obwohl nicht klar ist, ob diese zu einem späteren Zeitpunkt von Nutzen sind. Eine Beschränkung des Suchraums durch Restriktionsstrategien (siehe Unterkapitel 1.3.3) ist daher unbedingt angezeigt, um eine Vielzahl irrelevanter Klauseln gar nicht erst zu erzeugen. Allerdings geht das möglicherweise auf Kosten einer Beweisverlängerung: es kann – im Gegensatz zur Stufensättigung – nicht garantiert werden, dass der kürzeste Beweis gefunden wird (Stanford Encyclopedia 2004).

6.2 Ein konventioneller automatischer Beweiser – OTTER

Stellvertretend für alle konventionellen automatischen Beweiser wird hier das bekannte von McCune (1990) veröffentlichte System *OTTER* (die Abkürzung steht für engl. organized techniques for theorem proving and effective research) vorgestellt. Das durch eine Beweisaufgabe gegebene Problemwissen wird in der Vorbereitung auf vier Teilbereiche aufgeteilt:

1. Die *Unterstützungsmenge* `sos` enthält die wichtigen Fakten des Problems. Jeder Resolutionsschritt beinhaltet eine dieser Klauseln, so dass die Suche auf die Menge `sos` fokussiert wird.

2. Die Menge der *Gebrauchsaxiome* `use` legt das Hintergrundwissen fest.

3. *Demodulatoren*. Sie definieren Basisumformungen zur schnellen Vereinfachung hergeleiteter Klauseln. Es wird dabei keine Resolution angewendet.

4. Parametermenge. Es wird die Suchheuristik festgelegt; des Weiteren eine Filterfunktion, mit der uninteressante Klauseln von der Suche ausgeschlossen werden.

Diese vier Mengen muss der Benutzer von Hand bestimmen. Im Prinzip resolviert OTTER fortwährend eine Klausel aus `sos` mit einer Klausel aus der Menge `use`. Dazu wird in jedem Schritt eine `sos`-Klausel *K* in die Menge `use` verschoben. Anschließend wird *K* mit jeder anderen Klausel aus der Menge `use` resolviert und die Ergebnisse nach Vereinfachung und Filterung wiederum der Menge `use` zugeführt. Wie Klausel *K* aus der Menge `sos` ausgewählt wird, bestimmt die Heuristik. Üblicherweise ist ein Bestandteil eine Vorsortierung nach „Gewicht“, d. h. nach Größe, Schwierigkeit oder Ähnlichem. Es wird daher eine Art von Unit-Resolution (siehe Unterkapitel 1.3.3) präferiert.

OTTER terminiert, wenn entweder die leere Klausel abgeleitet werden konnte oder die Unterstützungsmenge geleert wurde. Der Vereinfachungsprozess des Resolutionsergebnisses umfasst neben den angesprochenen Demodulatoren auch die Verschmelzung gleicher Literale innerhalb einer Klausel sowie die Entfernung von Tautologien.

6.3 Erweiterung von *RESAC* zum automatischen Beweiser

Durch Molekültyp und Reaktionsvorschrift ist *RESAC* prädestiniert für den Einsatz als automatischer Beweiser. Die Objekte dieser Künstlichen Chemie haben nach Konstruktion die Struktur von Klauseln der Prädikatenlogik 1. Ordnung. Die angewendete Reaktionsvorschrift ist die Resolution (siehe Kapitel 3), so dass ein geeigneter Inferenzprozess auf Klauseln zur Verfügung steht. Im Folgenden sind nun die Erweiterungen beziehungsweise Parameter beschrieben, die *RESAC* zum automatischen Theorembeweiser machen.

1. Neben der Resolution muss auch die Inferenzregel Faktorisierung (siehe Def. 1.2.13) zum Einsatz kommen, ohne die das Beweiskalkül nicht vollständig wäre. Die Integration geschieht standardmäßig, kann aber vom Benutzer ausgeschaltet werden (sozusagen als Restriktionsstrategie).

2. Die Größe des Reaktors wird durch einen weiteren Parameter vom Benutzer festgelegt.

3. Gemäß den Ausführungen in Unterkapitel 1.3.3 können vom Benutzer Resolutionsstrategien vereinbart werden. Die voreingestellte Strategie der negativen Resolution mit Unterstützungsmengen erlaubt eine Reaktion beispielsweise nur dann, wenn erstens eine der Reaktionspartner ausnahmslos aus negativen Komponenten (Literale) besteht und zweitens eine der Reaktionspartner Unterstützung hat.

   Without strategy, a reasoning program will drown in new information

   Wos, Overbeek, Lusk und Boyle (1984, Seite 460)

4. Der Benutzer kann die Höchstgrenze von durchführbaren Resolutionen pro Klausel festlegen. Diese Maßnahme modelliert z. B. den Energieverlust des Materials durch Reaktion. Außerdem kann die Länge einer Klausel begrenzt werden. Oberhalb dieser Grenze werden Klauseln nicht mehr als stabil angesehen. Gezählt werden die nicht-logischen Symbole.

5. Tautologien werden ausgefiltert. Das beeinträchtigt die Vollständigkeit des Kalküls nicht.
6. Gleichlautende Literale werden ohne Einschränkung der Deduktion verdichtet.
7. Ein *RESAC*-Experiment wird beendet, wenn die leere Klausel als Zeichen eines hergeleiteten Widerspruchs produziert wird; der Beweis ist erbracht. Zum Zweiten wird ein Limit für die verstreichende Experimentalzeit festgelegt.
8. Die Menge *Startklauseln*, die die Befüllung des Reaktors steuert, enthält die durch Konvertierung der Problembeschreibung hervorgehenden Klauseln.
9. Beide von *RESAC* unterstützten Ersetzungsmethoden sind einsetzbar (eine Diskussion findet sich in Kapitel 4).
10. Der gewünschte Zufluss wird als Parameter durch den Benutzer spezifiziert. Theoretisch ist ein Zufluss unbedingt notwendig, da man zu keinem Zeitpunkt weiß, wie oft eine Klausel im Resolutionsbeweis Anwendung findet. Praktisch jedoch liefern auch geschlossene Reaktoren ausreichender Größe Beweise.

Punkte 3 bis 6 dienen der Einschränkung des Suchraums. Dies ist vor allem in der Simulation eine notwendige Maßnahme.

6.4 Prolog und *RESAC*

Prolog (Clocksin und Mellish 1981) ist die bei weitem meist benutzte Sprache der logischen Programmierung. Diese Programmiertechnik basiert auf der Umsetzung einer in einer formalen Sprache gegebenen Spezifikation in ein lauffähiges Programm. Dieses Programm besteht aus fortwährend durchgeführten Inferenzprozessen auf dem gegebenen Wissen, welches sich zusammensetzt aus gegebenen Fakten und einem Ziel. Prolog wird gerne eingesetzt beim Rapid-Prototyping, bei der Analyse (engl. parsing) natürlicher Sprache und in Expertensystemen mit vielfältigen Anwendungsbereichen. In diesem Unterkapitel liegt das Augenmerk auf dem automatischen Beweisen, einem weiteren Anwendungsbereich, in dem Prolog zum Einsatz kommt.

Prolog-Programme bestehen aus Mengen von Logik-Klauseln über einer eingeschränkten Prädikatenlogik. Die Ausführung geschieht von der Zielklausel ausgehend rückwärts verkettet (*Rückwärtsverkettung*, engl. backward chaining). Dabei wird versucht, die Bedingungen zu erhärten, die bei Erfüllung das Ziel implizieren. Nach Muster der Tiefensuche (engl. depth-first search) werden dazu erneut die vorgelagerten Bedingungen geprüft. Ein in die Tiefe gehender Suchvorgang bricht mit einer Unifikation ab, sobald eine Bedingung mit einem Faktum aus der Datenbasis instanziiert werden kann. Kann eine Bedingung nicht durch Fakten unter den bisher angesammelten Unifikatoren erhärtet werden, setzt ein Rückschritt (engl. backtracking) im Lösungsprozess ein. An einem schon vorher erreichten Punkt können nun andere Klauseln benutzt werden und zu anderen Unifikatoren führen. Im ganzen Prozess werden die Klauseln in der Reihenfolge ihres Auftretens in der Datenbasis zur Inferenz gebracht.

Betrachtet man den durch die Prolog-Ausführung aufgestellten Tiefensuchbaum der Problemlösungsfindung (*Problembaum* genannt), so fällt auf: Die Abarbeitung dieses Problembaums erfolgt

deterministisch. Die Reihenfolge der Klauseln in der Datenbasis, die als Programm angesehen werden kann, haben entscheidenden Einfluss auf die Effizienz; Russell und Norvig (2003) sprechen von der Achillesferse Prolog's. Die Autoren geben ein Beispiel für die Entstehung unendlicher Problembäume[1], die nur von einer Umordnung der Zeilen (in diesem Fall handelt es sich um Regeln) des logischen Programms herrühren. Prolog muss daher als unvollständiger Problemlöser angesehen werden (obwohl die zugrunde liegende SLD-Resolution für die hier relevante, eingeschränkte PL1 vollständig ist).

Des Weiteren führt Prolog redundante Inferenzen aus. Wiederum Russell und Norvig (2003) erwähnen Fälle, in denen die Zahl der Inferenzen exponentiell in der Zahl der Grundterme ist. Verwendete man statt der Rückwärtsverkettung eine Vorwärtsverkettung (engl. forward chaining), bei der zuerst kleine Teilprobleme gelöst werden, aus diesen Lösungen dann größere Teilprobleme erarbeitet werden, so fielen nur quadratisch viele Inferenzen an.

Die im zweiten Abschnitt angesprochene Einschränkung der Prädikatenlogik 1. Ordnung betrifft die Polaritäten der in einer Klausel vorkommenden Literale. Prolog verwendet definite Horn-Klauseln, das sind Klauseln mit genau einem positiven Literal. Diese Einschränkung ist eine Konzession, um logische Programmierung effizient genug zu machen. Das geht zu Lasten der Ausdrucksstärke. Es gibt Probleme, die nicht in dieser eingeschränkten Form der PL1 formuliert werden können, wohl aber in der uneingeschränkten. Das in Kapitel 6.8 vorgestellte *Steamroller*-Problem ist ein solches; es zeigt, dass solche Probleme nicht nur theoretischer Natur sind und kompliziert konstruiert werden müssten. Eine weitergehende Diskussion findet sich in (Nebel 1995, Aufgabe des Optimalitätsanspruchs).

Zur Steigerung der Effizienz verzichtet Prolog auf einen weiteren elementaren Bestandteil der logischen Deduktion, den so genannten *Occurcheck*[2]. Dieser wird nötig im Rahmen des Unifikationsalgorithmus. Nicht-korrekte Inferenzen sind so möglich, das System ist inkonsistent. Einerseits rechtfertigen die vielen Anwendungsmöglichkeiten von Prolog diese Praxis, andererseits macht dies die Anwendung von Prolog beim automatischen Beweisen mathematischer Probleme erst einmal unmöglich. Spezielle Erweiterungen sind hier notwendig.

RESAC bietet demgegenüber einen nicht-deterministischen Ansatz, der die Reihenfolge der Regeln und Fakten in der Wissensbasis (gebildet durch die Menge der *Startklauseln*) beliebig werden lässt. *RESAC* arbeitet mit der kompletten Prädikatenlogik 1. Ordnung und konserviert so ihre vollständige Aussagekraft. Die Inferenzen sind darüber hinaus zu jedem Zeitpunkt konsistent. Ohne Zusatzmechanismen sind alle in der PL1 formulierbaren Probleme direkt lösbar. Beim Lösungsprozess wird nicht nur eine einzige Strategie angewandt, z. B. Rückwärtsverkettung, sondern Probleme lösen sich i.A.[3] ohne vorgeschriebene Richtung. Es werden sowohl „top-down" als auch „bottom-up" Inferenzen durchgeführt. Das Effizienzproblem zu umgehen ist das Ziel des Wechsels zu einem anderen Berechnungsparadigma. Nach Vorbild massiv parallel arbeitender natürlicher Systeme benutzt *RESAC* den Rahmen der Künstlichen Chemien zur Formulierung eines automatischen Beweisers, der auf einer Vielzahl elementarer Einzelreaktionen beruht.

[1] Prolog terminiert nur aufgrund einer vom Benutzer anzugebenden DFS-Suchbaumtiefe.

[2] Test bei der Substitution, ob innerhalb eines einer Variablen zugeordneten Terms diese erneut auftritt (Def. 1.2.10).

[3] Der Nutzer bekommmt jedoch die Möglichkeit, durch die Unterstützungsmengen-Restriktion Einfluss zu nehmen.

6.5 Einfacher Beweis in *RESAC*

Mit den angesprochenen Modifikationen ist das System *RESAC* nun in der Lage, wie ein automatischer Theorembeweiser Beweise zu finden. Ein einführendes Beweisproblem demonstriert das grundsätzliche Verhalten. Die natürlichsprachliche Beschreibung ist (Gini 1995) entnommen und in Tabelle 6.1 dargestellt; sie führt nach Umwandlung in Klauselform zu acht prädikatenlogischen Klauseln[4]. Der von der Künstlichen Chemie durch Resolutionsinferenz gefundene Beweis wird ebenfalls in Tabelle 6.1 vorgestellt. In diesem Beweis werden die abgeleiteten Klauseln zunächst immer länger, bevor sie sich wieder verkürzen, und letztlich die leere Klausel abgeleitet wird.

Aufgabe:	**Resultierende Problemklauseln (formen die Menge *Startklauseln*):**
(1) All dogs howl at night.	s1. ¬dog(x), howls(x)
(2) Anyone who has any cats will not have any mice.	s2. ¬has(x,y), ¬cat(y), ¬has(x,z), ¬mice(z)
(3) Light sleepers do not have anything which howls at night.	s3. ¬lightsleep(x), ¬has(x,y), ¬howls(y)
(4) John has either a cat or a dog.	s4. has(John,S)
(Ziel) If John is a light sleeper, then John does not have any mice.	s5. cat(S) dog(S)
	s6. lightsleep(John)
	s7. has(John,M)
	s8. mice(M)

Von *RESAC* gefundener Beweis:

1.	cat(S), howls(S)	s5 und s1
2.	cat(S), ¬has(x,S), ¬lightsleep(x)	1 und s3
3.	¬has(x,S), ¬has(y,z), ¬has(y,S), ¬lightsleep(x), ¬mice(z)	2 und s2
4.	¬has(x,y), ¬has(x,S), ¬has(John,S), ¬mice(y)	3 und s6
5.	¬has(x,M), ¬has(x,S), ¬has(John,S)	4 und s8
6.	¬has(John,S)	5 und s7
7.	leere Klausel	6 widerspricht s4.

Tabelle 6.1: Einfaches Beweisproblem (Gini 1995). Lösung durch die Künstliche Chemie *RESAC*.

Aspekt Reaktorgröße Neben der Komplexität eines Problems hat die Größe des Reaktors entscheidenden Einfluss auf die Zahl der Kollisionen, die zur Beweisfindung benötigt werden. Die Einfachheit des hier gegebenen Problems ermöglicht nun eine Studie dieses Aspekts mit Hilfe einer großen Zahl von Experimentdurchläufen. Abbildung 6.1 zeigt annähernd 10000 Läufe. Dabei wurden unterschiedliche Zeitvorgaben und Reaktorgrößen untersucht. Die Zahl der in der vorgegebenen Zeit erfolgreichen Läufe, d. h. einen Beweis hervorbringende Läufe, werden protokolliert. Es wird deutlich, dass ein Reaktor mit mehr Molekülen signifikant mehr Kollisionen benötigt.

Wählt man andererseits die Reaktorgröße zu klein, kann das eine Beweisfindung nicht nur verzögern, sondern ganz und gar verhindern (siehe Abb. 6.2). Je komplexer ein Beweis ist, desto mehr Zwischenschritte müssen im Reaktor vorgehalten oder über Reaktionspfade nachgebildet werden. Darüber hinaus wirkt es beschleunigend, von wichtigen Klauseln eine gewisse Anzahl zur Hand zu haben, um schnell ein breites auf dieser Klausel basierendes Reaktionsnetz zu entfachen.

[4] Aufgrund der Klausel s5 ist diese Klauselmenge nicht durch Horn-Klauseln darstellbar

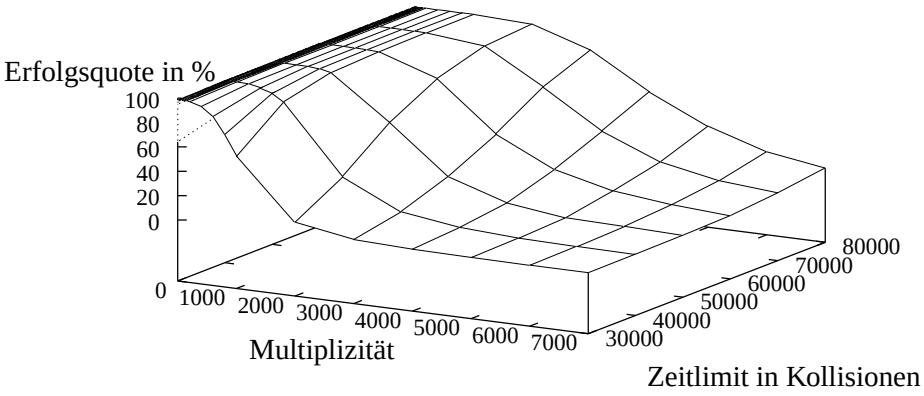


Abbildung 6.1: Anzahl erfolgreicher Experimente zu verschiedenen Reaktorgrößen. Zum einfachen Beweisproblem aus Tabelle 6.1 wird die erreichte Erfolgsquote zu variierender Multiplizität und Zeitvorgabe (in Kollisionen) aufgetragen. Jeder Punkt repräsentiert 50 Experimentdurchgänge im Zuflussreaktor (elastischer Zufluss). Erklärung siehe Text.

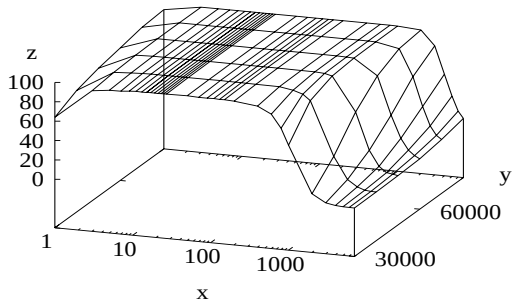


Abbildung 6.2: Anzahl erfolgreicher Experimente zu verschiedenen Reaktorgrößen. Das Bild zeigt noch einmal den Graphen aus Abb. 6.1. Die x-Achse ist diesmal logarithmiert dargestellt. Sie zeigt die Multiplizität. Die y-Achse gibt die Zahl der jeweils erlaubten Kollisionen an. Die erreichte Erfolgsquote (in %) zeigt die z-Achse. Ein kleiner Reaktor kann die Beweisfindung verhindern.

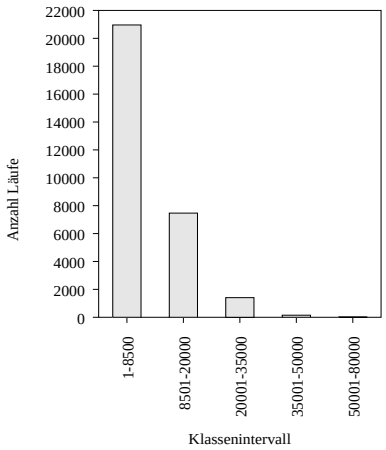


Abbildung 6.3: Einfaches Beweisproblem aus Tabelle 6.1. Klassifikation der zum Beweis benötigten Kollisionen. 30000 Läufe wurden durchgeführt. Numeriert man die Klassen von null (1-8500 Kollisionen) an, so ist der Mittelwert dieser Poisson-Verteilung $\mu = 0,3601$. Elastischer Zufluss, Multiplizität 20.

Experimentalzeit Wird *RESAC* als Beweiser verwendet, kann die Laufzeit eines Experiments durch die Zahl der für eine Beweisfindung erforderlichen Kollisionen gemessen werden. Bildet man Laufzeitklassen, so entspricht die Verteilung der benötigten Kollisionen einer Poisson-Verteilung (Abbildung 6.3). Im Allgemeinen hängt der Mittelwert der Verteilung von den Experimentparametern (hauptsächlich fällt die Reaktorgröße ins Gewicht) und Problemkomplexität ab.

6.6 Beweisproblem aus der Gruppentheorie

Das folgende Problem ist ein Standardbeispiel aus dem Bereich der Mathematik. Bei gegebenen Gruppenaxiomen soll bewiesen werden, dass jedes Element eine Rechtsinverse hat (siehe Tabelle 6.2, Seite 116). *RESAC* findet einen Beweis der Länge 8. Wiederum ist die Zahl der benötigten Kollisionen poissonverteilt.

Problemklauseln:

(1) P(X,Y,f(X,Y))
(2) ¬P(X,Y,U), ¬P(Y,Z,V), ¬P(X,V,W),P(U,Z,W)
(3) ¬P(X,Y,U), ¬P(Y,Z,V), ¬P(U,Z,W),P(X,V,W)
(4) P(e,Y,Y)
(5) P(g(Y),Y,e)
(6) ¬P(X,h(X),h(X)), ¬P(k(X),Z,X)

Tabelle 6.2: Problem aus der Gruppentheorie. Existenz der Rechtsinversen in jeder Gruppe für jedes Element nach (Loveland 1978). f, h, k sind Skolemfunktionen.

Problemklauseln:

s1. f(e,X)=X
s2. f(X,e)=X
s3. f(g(X),X)=e
s4. f(X,g(X))=e
s5. f(X,f(Y,Z))=f(f(X,Y),Z)
s6. X=X
s7. X≠Y, Y=X
s8. X≠Y, Y≠Z, X=Z
s9. U≠W, f(U,X)=f(W,X)
s10. U≠W, f(X,U)=f(X,W)
s11. U≠W, g(U)=g(W)
s12. f(f(X,X),X)=e
s13. h(X,Y)=f(f(f(X,Y),g(X)),g(Y))
s14. h(h(a,b),b)≠e

Tabelle 6.3: Burnside-Problem der Ordnung 3 (M. Hall 1968).

Makroskopisches Verhalten Abbildung 6.4 untersucht die dynamischen Prozesse in verschiedenen Experimenten zu diesem Beweisproblem. Es kommen dabei einige der in Kapitel 2 eingeführten makroskopischen Indikatoren zum Einsatz. Verglichen wird ein geschlossener Reaktor mit einem Zuflussreaktor. Gezeigt wird hier der elastische Zufluss, bei dem genau dann ein Molekül dem Reaktor zufließt, wenn die vorhergehende Reaktion elastisch war. Der Zufluss wird so durch die Produktivitätsrate gesteuert. Mit dieser Zuflussstrategie verbindet sich die Hoffnung, adäquat auf eine Veränderung der Systemaktivität reagieren zu können. Einer Blockade des Systems durch Inaktivität wird durch „Verdünnung“ des Reaktorinhalts entgegengewirkt. In der Tat kann so die Produktivitätsrate auf einem Niveau gehalten werden, ganz im Gegensatz zum geschlossenen Reaktor, in dem die Produktivität im Laufe der Zeit deutlich absinkt (siehe erneut Abb. 6.4). Erkauft wird sich die relativ hohe Aktivität durch einen hohen Anteil von Startklauseln, der schon angesprochenen Verdünnung. Die Diversität in solchermaßen geöffneten Reaktoren ist daher deutlich geringer als in geschlossenen Systemen, in denen die anfänglich hohe Zahl erfolgreicher Reaktionen zu vielen neuen Klauseln führt (Diversifikationsphase), die die Startklauseln nach und nach ersetzen. Die spezialisierten (ausgeformten) Klauseln weisen nicht mehr die Reaktionsfähigkeit der in ihrer Form allgemeineren Startklauseln auf; das geschlossene System befindet sich in der Gefahr blockiert zu werden. Abbildung 6.4 zeigt auch, dass die jeweils 100 durchgeführten Experimente sehr ähnliche makroskopische Charakteristika aufweisen. Offene Reaktoren fanden den Beweis durchweg in kürzerer Zeit.

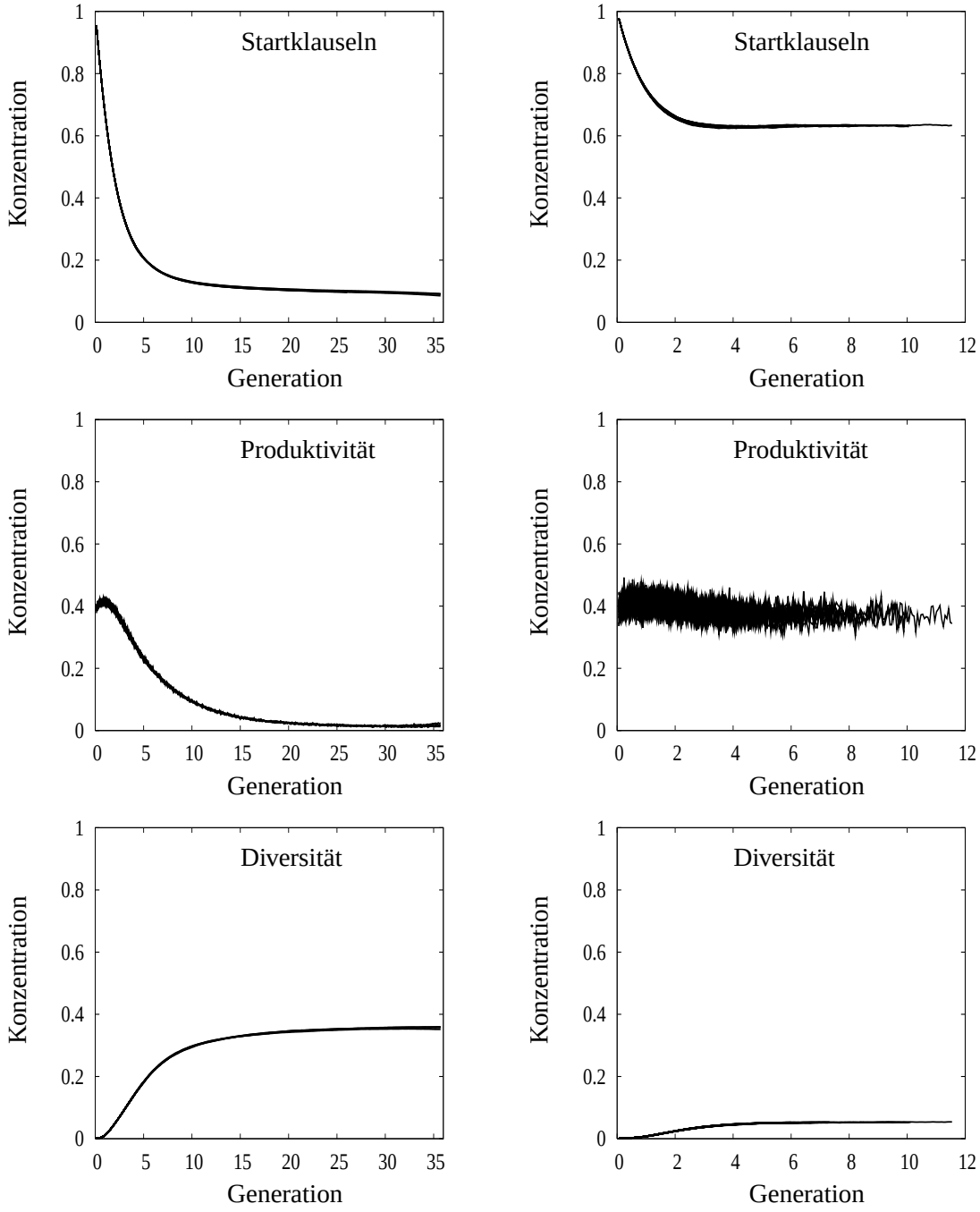


Abbildung 6.4: Anteil der Startklauseln, Produktivität und Diversität in 100 Experimenten zum Problem aus Tabelle 6.2. Multiplizität $\mu = 14000$. Alle 100 Experimente sind in jeweils einen Graphen gezeichnet. Sehr ähnliche Kurvenverläufe sind zu beobachten. Allerdings sind die Kurven unterschiedlich lang. Die zum Beweis nötige Zeit variiert. *Links:* Geschlossener Reaktor. *Rechts:* Reaktor mit elastischem Zufluss.

6.7 Burnside-Problem

Das so genannte Burnside-Problem für Gruppen der Ordnung drei (M. Hall 1968) hat eine gewisse Tradition im Bereich des automatischen Beweisens. Es weist eine hohe Komplexität auf, zumindest wenn als Kalkül nur das Prädikatenkalkül mit der Inferenzregel Resolution verwendet wird.[5] Robinson und Wos (1969) zeigen einen Beweis, der über 138 Zwischenschritte führt.

Das Burnside-Problem verlangt nach einem Beweis, dass in allen Gruppen der Ordnung drei gilt:

$$x^3 = e \longrightarrow [[x,y],y] = e,$$

wobei e das Einselement und der Kommutator $[x,y]$ das Element $xyx^{-1}y^{-1}$ bezeichnet. Tabelle 6.3 zeigt die Problemklauseln. Sie bestehen neben dem Beweisziel im Wesentlichen aus den Gruppenaxiomen. Die Eigenschaften der Gleichheitsrelation, die hier als Prädikat aufgenommen ist, müssen einzeln spezifiziert werden.

Erfolgsquote In der Computersimulation von *RESAC* lässt sich kein Beweis herleiten. Ein Grund liegt in der verkomplizierenden Behandlung der Gleichheitsrelation. Die in der Beweisformulierung auftauchenden sechs Gleichheitsaxiome sind alle notwendig, um die Gleichheitsrelation für das Prädikat „=“ zu axiomatisieren. Sie führen zu einer enormen Vergrößerung des Lösungsraums, denn einerseits kann jede abgeleitete Aussage mit fast allen diese Axiomen resolviert werden, andererseits bilden die Gleichheitsaxiome auch untereinander ein Reaktionsnetzwerk aus, welches in der Theorie immer weiter wächst. Man betrachte als Beispiel das Axiom s8: $(X = Y \wedge Y = Z) \rightarrow X = Z$. Durch autokatalytische Reaktion entsteht ein Theorem, das die Transitivität über zwei Zwischenelemente beweist. Diese Klausel kann wiederum mit sich selbst oder s8 resolvieren. Durch fortlaufende Reaktionen nur unter s8 und Abkömmlingen entstehen immer längere Transitivitätstheoreme. Sie sind in der Praxis nur durch die in *RESAC* geltende Molekülängenrestriktion begrenzt (siehe Unterkapitel 6.3).

Der Suchraum weist nach Unterkapitel 6.1 möglicherweise exponentielles Wachstum auf. Die für diese Experimente zur Verfügung stehenden Ressourcen reichten für die Herleitung eines Beweises nicht aus. Es wird bei der Lösungssuche angesichts der Suchraumgröße ganz besonders darauf ankommen, wie sich das Problem gegenüber bestimmten Suchraumrestriktionen, also Steuerungsstrategien (siehe Unterkapitel 1.3.3) verhält. Da beim System *RESAC* aber die problemunabhängige Herangehensweise im Vordergrund steht und die Komplexität der Reaktionsdurchführung beschränkt sein soll (im Kapitel 8 wird darauf zurückgekommen), ist diese Künstliche Chemie den Spezialisten für automatisches Theorembeweisen in der Anwendung von mehreren, zum Teil verkoppelten Heuristiken unterlegen. Bei einer (zukünftigen) Übertragung von *RESAC* auf natürliche Systeme mit einer ungeheuren Vielzahl von parallel durchgeführten Reaktionen sieht die Sache dagegen anders aus.

Paralleles Experiment Auch durch die lineare Zusammenschaltung (siehe Unterkapiel 3.3) parallel arbeitender Reaktoren konnte kein Beweis gefunden werden. Abbildung 6.5 zeigt die Pro-

[5] Robinson und Wos (1969) benutzten gerade dieses Problem, um zu zeigen, dass die Verwendung der Paramodulation zu einer enormen Erleichterung führt.

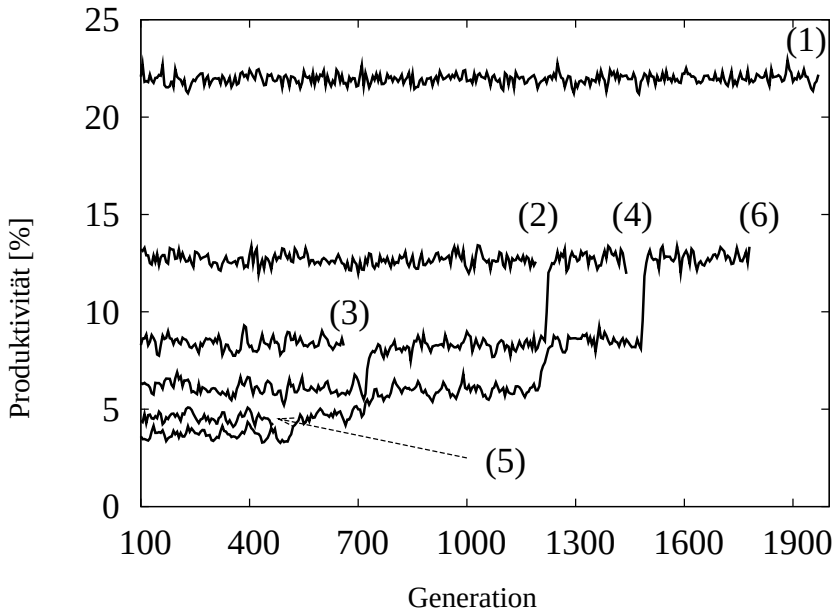


Abbildung 6.5: Produktivität in 6 parallel arbeitenden Reaktoren im Burnside-Experiment. Die Reaktoren sind linear angeordnet und der Abfolge nach mit (1),. . .,(6) bezeichnet, wobei Reaktor (1) der erste Reaktor der Kaskade ist. Gezeigt ist die Produktivität in einem Experiment mit freier Ersetzung und Zuflussrate $i = 0,1$. Jeder Reaktor beinhaltet 14000 Moleküle. In den ersten Reaktor (1) fließen zu 100% Startklauseln. Die folgenden Reaktoren haben eine von ihrer Position in der Reaktorkette abhängige Produktivität. Nacheinander werden die Reaktoren (5), (3), (2), (4) und (6) abgeschaltet. Die jeweils nachfolgenden Reaktoren verringern den Abstand zum 1. Reaktor und bilden eine Produktivität aus, die der neuen Position in der Kette entspricht.

duktivität in einer solchen Reaktorkette. Bei der linearen Anordnung fließen ausschließlich Startklauseln in den ersten Reaktor, der das Anfangsglied der Kette bildet. Er arbeitet daher wie ein isolierter Reaktor. Für die Startklauseln stellt sich ein problemspezifisches Konzentrationsniveau ein, bei diesem Problem sind durchschnittlich 33% aller Klauseln Startklauseln. Dieses bestimmt maßgeblich die Produktivität, denn die Startklauseln in diesem Problem sind mehrheitlich unspezifisch, d. h. sie führen oft zu produktiven Reaktionen. Es sind die Gleichheitsaxiome, die bei vielen Kollisionen zur Reaktion führen.

Für die folgenden Reaktoren gilt, dass der Zufluss bestimmt wird durch die Molekülkonzentrationen im Vorgängerreaktor. Dem zweiten Reaktor fließt so ein Gemisch zu, in dem nur noch zu ungefähr 33% Startklauseln vertreten sind. Eine niedrigere Produktivität ist die Folge. Durch die Reaktionen in diesem Reaktor stellt sich wiederum ein Gleichgewicht an Startklauseln ein, hier ungefähr 15%. Dem 3. Reaktor fließt so auch nur ein Startklausel-Anteil in gleicher Höhe zu.

Fällt ein Reaktor in der Kaskadierung aus, so übernimmt der nächstfolgende seine Position. Er hat – wie alle nachfolgenden Reaktoren – seinen Abstand zum 1. Reaktor um eine Position verringert. Der Zufluss enthält nun mehr Startklauseln. Die Produktivität steigt auf das Niveau, das schon der ausgefallene Reaktor aufwies.

6.8 Schubert's Steamroller

Folgende Denksportaufgabe stellte Lenhart Schubert 1978 den zu seiner Zeit bestehenden Theorembeweisern:[6]

> Wölfe, Füchse, Vögel, Raupen und Schnecken sind Tiere, und von jeder der aufgezählten Arten gibt es einige Exemplare; außerdem gibt es Getreidesorten, wobei Getreidesorten Pflanzen sind. Jedes Tier frisst entweder alle Arten von Pflanzen oder alle sehr viel kleineren Tiere, die Pflanzen fressen. Raupen und Schnecken sind sehr viel kleiner als Vögel, die sehr viel kleiner als Füchse sind. Diese wiederum sind sehr viel kleiner als Wölfe. Wölfe fressen weder Füchse noch Getreidesorten, während Vögel Raupen, aber keine Schnecken fressen. Raupen und Schnecken fressen Pflanzen. Daher gibt es ein Tier, das ein getreidefressendes Tier frisst.

Stimmt die Behauptung, und wenn ja, um welches Tier handelt es sich? Aufgrund seiner enormen kombinatorischen Komplexität wurde dieses Problem unter dem Namen „Schubert's Steamroller" berühmt. Eine ausführliche Analyse findet sich in (Stickel 1986). Es stellt sich heraus, dass dieses Problem nicht in Hornlogik, einer eingeschränkten Prädikatenlogik 1. Ordnung wie beispielsweise Prolog sie benutzt, lösbar ist.

Zur Lösung wird der Problemtext ganz systematisch Satz für Satz in die Syntax der PL1 übersetzt. Ein sich aus dem ersten Problemsatz ergebender prädikatenlogischer Satz ist der folgende:

$$\forall X\,(\mathrm{Wolf}(X) \rightarrow \mathrm{Animal}(X)) \;\wedge\; \exists X\,\mathrm{Wolf}(X). \tag{6.1}$$

Der sich aus dem 2. Problemsatz ergebende PL1-Satz ist

$$\begin{aligned}&\forall X\,(\mathrm{Animal}(X) \rightarrow \\ &\quad \forall Y\,(\mathrm{Plant}(Y) \rightarrow \mathrm{Eats}(X,Y)) \vee \\ &\qquad \forall Z(\mathrm{Animal}(Z) \wedge \mathrm{Muchsmaller}(Z,X) \wedge\; \exists U\,(\mathrm{Plant}(U) \wedge \mathrm{Eats}(Z,U)) \rightarrow \mathrm{Eats}(X,Z))).\end{aligned} \tag{6.2}$$

Es ergeben sich in der Fortführung 17 prädikatenlogische Sätze, die die Problembeschreibung P bilden. Schließlich wird der letzte Problemsatz, der die Folgerung F beinhaltet, in PL1 umgesetzt:

$$\exists X\,\exists Y\;(\,\mathrm{Animal}(X) \wedge \mathrm{Animal}(Y) \wedge \mathrm{Eats}(X,Y) \;\wedge\; \exists Z\,(\mathrm{Grain}(Z) \wedge \mathrm{Easts}(Y,Z))\,).$$

Um die Folgerung zu bestätigen, wird dieser negiert und versucht, zusammen mit P einen Widerspruch herzuleiten (siehe Kapitel 1). Im zweiten Vorbereitungsschritt werden die prädikatenlogischen Sätze zur Anwendung der Resolution in Klauselform (siehe Unterkapitel 1.2.2) konvertiert. Dabei werden auch einstellige Skolemfunktionen eingeführt. Aus (6.1) entstehen beispielsweise folgende Klauseln (mit Skolemkonstante w):

$$\left\{\overline{\mathrm{Wolf}(X)},\, \mathrm{Animal}(X)\right\},$$
$$\{\mathrm{Wolf}(w)\}.$$

[6] aus dem Englischen übersetzt

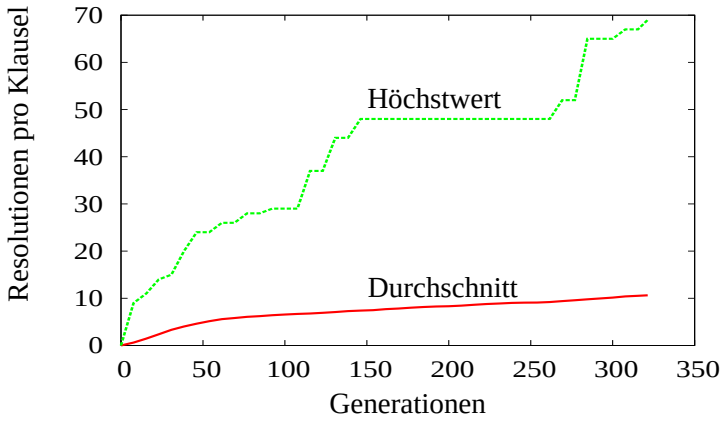


Abbildung 6.6: Resolutionsentwicklung im Steamroller-Experiment. Gezeigt ist ein erfolgreicher Lauf in *RESAC*, freie Ersetzung und 0,1%-Zufluss. Der Zufluss ist gerade so schwach, dass im Durchschnitt eine stetige Erhöhung der Zahl der Resolutionen pro Klausel erlaubt wird, andererseits aber so stark, dass die Startklauseln und darauf aufbauende Reaktionspfade nicht versiegen. Neue Klauseln führen zu neuen Resolutionshöchstwerten.

Einen interessanten Aspekt zeigt die Umwandlung des Satzes 6.2:

$$\overline{\text{Animal}(X)}, \overline{\text{Plant}(Y)}, \overline{\text{Animal}(Z)}, \overline{\text{Muchsmaller}(Z,X)}, \overline{\text{Plant}(U)}, \overline{\text{Eats}(Z,U)}, \\ \text{Eats}(X,Y), \text{Eats}(X,Z). \quad (6.3)$$

Zu bemerken ist hier das Auftreten zweier positiver Literale innerhalb einer Klausel. Diese Klausel entspricht so nicht der Syntax der Horn-Klauseln (siehe Def. 1.2.9). Damit ist diese Aufgabe nicht durch Beweiser (wie Prolog) lösbar, die auf Basis von Horn-Klausel arbeiten. Letztlich wird noch die Folgerung F in Klauselform angegeben:

$$\overline{\text{Animal}(X)}, \overline{\text{Animal}(Y)}, \overline{\text{Grain}(Z)}, \overline{\text{Eats}(Y,Z)}, \overline{\text{Eats}(X,Y)}.$$

Insgesamt werden 26 Klauseln aufgestellt. Schon unter diesen gibt es genau 87 Möglichkeiten, Resolutionen anzuwenden. Die Zahl der möglichen Schnitte wächst schlimmstenfalls exponentiell (vgl. Unterkapitel 6.1) und es sind mindestens acht Resolutionen nötig, um einen Widerspruch abzuleiten wie aus Satz 6.3 hervorgeht. Es ist nach Hofbauer und Kutsche (1991) im vorneherein für einen automatischen Theorembeweiser nicht zu erkennen, welche Klauseln wichtig bzw. welche Schnittfolge erfolgversprechend ist. Raupen beispielsweise spielen bei diesem Beweis überhaupt keine Rolle. Auch ist es egal, dass Schnecken Tiere sind.

RESAC findet eine Lösung dieses Problems mit 69 Resolutionen in Generation 321 bei einer Reaktorgröße von 2600 Klauseln. Ohne Problemwissen, nur durch Resolution und Restriktionsstrategie weisen die im Reaktor befindlichen Klauseln im Durchschnitt eine mit der Zeit steigende Anzahl von durchgeführten Resolutionen auf (siehe Abb. 6.6). Die leere Klausel, Zeichen des gefundenen Widerspruchs, entstand durch Resolution der Klauseln $\text{Eats}(f,b)$ und $\overline{\text{Eats}(f,b)}$. Füchse sind also Tiere, die andere getreidefressende Tiere fressen, nämlich Vögel.

Kapitel 7

Programmierung von *RESAC*– Implementierung eines Entscheidungsproblems

Im Gegensatz zu anderen Künstlichen Chemien beinhaltet weder das Reaktionsschema von *RESAC* noch die Struktur der Moleküle problemspezifisches Wissen. Beide Systemkomponenten sind darüber hinaus von einer definierten und unveränderlichen Struktur. Wie in einem so konstruierten System Aufgabenlösungen implementiert werden können, zeigen die folgenden Kapitel.

7.1 Programmierung des Systems *RESAC*

In Teil II wurde *RESAC* vorgestellt als eine resolutionsbasierte Künstliche Chemie. Diese Chemie verwendet Strukturen des Prädikatenkalküls 1.Ordnung. Dabei bilden logische Klauseln die Menge der Objekte im Reaktor; die Technik der Resolution definiert das Interaktionsschema. Das prädestiniert *RESAC* für den Einsatz als automatischer Theorembeweiser. Neben dieser im vorangegangenen Kapitel dargelegten Anwendung ist es aber auch denkbar, das System *ohne* Interpretation einer Logik einzusetzen. Aus dieser Perspektive wird dann unter Verzicht der Semantik allein die klar definierte Syntax der Prädikatenlogik und der darauf definierte Reaktionsmechanismus der Resolution ausgenutzt. Während man die widerlegungsvollständige Beweisbarkeit verliert, gewinnt man konstruktive Freiheit, die es erlaubt, allgemeine Sachverhalte ungezwungener darzustellen und Probleme möglicherweise intuitiver zu formulieren. Diese Freiheit besteht nicht in der Relaxierung von syntax-bedingten Restriktionen, sondern in der Loslösung einer erfüllbarkeitstheoretischen Interpretation der Objekte des Diskurses, hier der Moleküle.

Welche Vorteile bietet die Beibehaltung der strukturbezogenen Komponenten des Kalküls? Zieht man sich auf die klar definierte Syntax der PL1 zurück, so kann man *ohne* problemspezifische Anpassung des Reaktionsschemas Anwendungen in einer KC formulieren, deren Algorithmus, Reaktionsregelsatz und Molekülstruktur fest vorgegeben ist. Die Folge ist ein einfaches, standardisiertes Verfahren zur KC-Konstruktion, das zu vereinfachten Analysen und vergleichbaren Systemen führt. Eine solche in sich invariabel aufgestellte KC ist trotzdem universell. Sie stellt gleichsam ein Programmiersystem dar. In *RESAC* ist die PL1 gewissermaßen die Programmiersprache.

Programm	Menge von Klauseln über dem Prädikatenkalkül
Instruktion gemäß der Programmiersprachen-Syntax	Implizit durch Klauselformulierung gegeben
Befehlsausführung	Anwendung der Resolution
Daten	Klauselinhalt
Speicher	Reaktor
Programmeingabe	Initialbefüllung des Reaktors und Reaktorzulauf
Programmausgabe	Variabel: Experimentabhängig

Tabelle 7.1: Gegenüberstellung der Strukturen: Traditionelle Programmiersprache und *RESAC*.

In diesem Bilde steht die Resolution für die Exekution einzelner Programminstruktionen. Da die Wirkungsweise der Resolution an sich fest vorgegeben ist, wird klar, dass die Programminstruktion hauptsächlich[1] durch die Klausel selbst definiert wird. Es ist die Anzahl, die Verknüpfung und die Wahl der Prädikate, Funktionen und Variablen, die Einfluss auf die Resolution und damit auf das Ergebnis haben. Das problemspezifische Moment liegt in der Ausgestaltung der Moleküle, sie leiten den Fortgang bei der Problemlösung durch die indirekte Steuerung der Resolution. Im Gegensatz zum Schema konventioneller Programmiersprachen bildet sich die Instruktion auch erst durch das Zusammenwirken mehrerer Klauseln (bei binärer Resolution sind es zwei).

In der resolutionsbasierten Chemie sind die Fakten eines Problems (Daten, Objekte) ebenfalls in den im Reaktor befindlichen Klauseln kodiert. Die Moleküle erfüllen in einer solchen Chemie also eine Doppelrolle: Sie sind sowohl Informationsträger als auch Instruktion. Instruktionen steuern in einer Künstlichen Chemie das *Zusammenwirken der Datenströme* und damit die Problemlösung. In herkömmlichen Programmiersprachen kann man dagegen von einer gezielten *Manipulation von Datenströmen* sprechen. Die Doppelrolle der Klauseln charakterisierend, stellt sich die Frage, wie die beiden Funktionen miteinander verkoppelt sind. Sie gehen Hand in Hand, sind parallel verzahnt, die Klausel bietet sozusagen einen aktiven Teil (die Richtungssteuerung) und einen passiven Teil (die Daten) an. Die Verarbeitung kann gleichzeitig erfolgen. Demgegenüber steht die üblicherweise strikte Unterscheidung von Daten und Anwendungsanweisungen in einer herkömmlichen Programmiersprache.[2] Daneben ist auch eine Doppelrolle anderer Art bekannt: In der Binärstring-Chemie von Banzhaf (1993) ist ein Molekül entweder Operator oder Operand (jedoch nicht beides gleichzeitig). Die Rolle des Moleküls kann sich bei jeder Reaktion ändern. Zu diesem Zweck wird dort eine geeignete Abbildung definiert. Will man das Äquivalent zum herkömmlichen Begriff des iterativen Programms benennen, muss auf die Gesamtheit aller im Reaktor vorhandenen und zusammenwirkenden Klauseln verwiesen werden.

Es wird deutlich, dass ein präziser, theoretisch angesetzter Paradigmenvergleich zwischen einer traditionellen Programmiersprache einerseits und einer resolutionsbasierten Chemie andererseits schwierig ist. Es ist aber keineswegs eine 1 : 1 Übersetzung zu erwarten, wie die gerade erfolgte

[1] Es werden später individuelle Mechanismen wie die *separate Multiresolution* eingeführt, die es erlauben, das makroskopische Resolutionsverhalten auf spezielle Problembereiche abzustimmen und das Gesamtresultat geeignet zu modifizieren.

[2] Daran ändert auch die Tatsache nichts, dass in der von-Neumann-Architektur Daten und Anweisungen im gemeinsamen Speicher stehen.

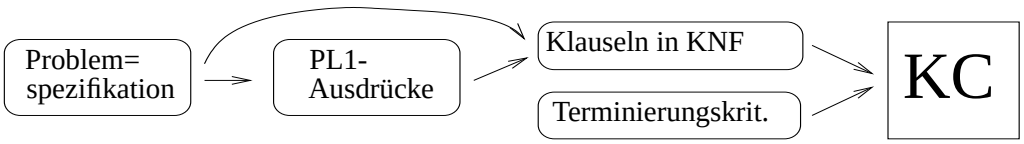


Abbildung 7.1: Konvertierungsschritte von der Problemspezifikation zur Eingabe von *RESAC*. Das separat aufgelistete Terminierungskriterium resultiert ebenfalls aus der Problemspezifikation.

Diskussion zeigt. Sehr deutlich hingegen sind die Parallelen zwischen den Programmiersystemen. Eine mögliche Sichtweise ist in Tabelle 7.1 zusammengefasst. *RESAC* ähnelt dagegen der logischen Programmierung. Logische Programmiersprachen wie etwa Prolog fassen ein Programm als logische Formel auf. Im Allgemeinen wird ein *Programmziel* aus einer Menge anderer Aussagen konstruktiv abgeleitet. Die Deduktion selbst kann als Berechnung aufgefasst werden (Hayes 1973). Das Resolutionsprinzip und die Unifikation ermöglichen eine automatische Abarbeitung einer logischen Spezifikation. Idealerweise ergibt sich nach Kowalski's bekannter Gleichung, dass ein Algorithmus aus Logik und Ablaufkontrolle besteht (Kowalski 1979a). Diese Denkweise ähnelt der Anschauung im System *RESAC*. Jedoch sind beide Komponenten unterschiedlich realisiert. Während in *RESAC* das volle Prädikatenkalkül 1. Ordnung Verwendung findet, wird diese bei der log. Programmierung eingeschränkt. Die Einschränkung führt nach (Bläsius und Bürckert 1992) zu einer höheren Effizienz, einer geringeren Komplexität der Ablaufkontrolle und der Determiniertheit des Programmablaufs. Nachteilig ist die Einschränkung der Ausdrucksfähigkeit und der starre Suchmechanismus. Im Unterkapitel 6.4 wird näher darauf eingegangen. Dort wird *RESAC* mit Prolog verglichen.

Nach den Vorüberlegungen ist klar, dass sich die Programmierung von *RESAC* auf eine geeignete Formulierung der Klauseln reduziert (Abb. 7.1). Dabei ist es einerseits möglich, die gegebene Problemspezifikation in die PL1 zu übersetzen und anschließend wie in Unterkapitel 1.2.2 gezeigt, in die konjunktive Normalform zu konvertieren. Die Klauseln haben dann direkt die von *RESAC* benötigte Eingabeform. Schließlich wird festgelegt, wann das Experiment beendet wird. Das Auftreten einer bestimmten Klausel oder eine Zeitvorgabe sind mögliche Terminierungskriterien. Probleme, die einfach durch ein Logikkalkül zu beschreiben sind (siehe Kap. 1.1.1: Konzeptualisierung), sind auf diese Weise schnell in *RESAC* zu implementieren. In Kapitel 6 wird dieser Ansatz verfolgt. So kann etwa die Berechnung von Funktionen (Fakultätsfunktion (Kowalski 1974)), das Sortieren[3] (Kowalski 1979b) oder Datenbankanfragen (Gallaire et al. 1984) in Logik dargestellt werden. Darüber hinaus zeigen Wos et al. (1984) die automatische Programmverifikation, Schaltungsentwurf und -analyse sowie Kontrolle von Echtzeitsystemen.

Andererseits kann aber auch von der Problemstellung direkt auf eine geeignete Klauseldarstellung geschlossen werden. Wie das funktioniert, wird im Folgenden anhand verschiedener Beispiele gezeigt. Es kommt dabei darauf an, eine gewisse Intuition zu entwickeln, wie bestimmte syntaktische Konstrukte des Prädikatenkalküls durch Anwendung der Resolution modifiziert werden beziehungsweise nur bestimmte Resolutionen zulassen. Eine eindeutig als „richtig" zu klassifizierende Umsetzung gibt es nicht (eine Facette der ontologischen Unverbindlichkeit der KI, siehe Unterkapitel 1.1.1), wohl aber zweckmäßige und weniger zweckmäßige.

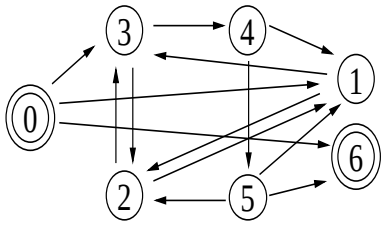
[3] Unter den verwendeten Verfahren befindet sich auch *Quicksort*.

7.2 Suche nach gerichteten Hamilton-Pfaden

Die Programmierung der Künstlichen Chemie *RESAC* soll in diesem Kapitel anhand der Lösung eines NP-vollständigen Entscheidungsproblems gezeigt werden. Dazu soll das berühmt gewordene so genannte Adleman-Problem nachvollzogen werden. Adleman (1994) verknüpfte dabei erstmalig die Gebiete der Molekularbiologie und der Informatik, um das gerichtete Hamilton-Pfad-Problem (abgekürzt DHPP von engl. directed Hamiltonian path problem) auf molekularer Ebene in Form von parallel arbeitenden DNA-Berechnungen zu lösen. Obwohl das Problem in der vorliegenden Dimension nicht sehr komplex ist, erscheint es geeignet zur Demonstration der Herleitung der Eingabeklauseln.

Beim DHPP ist ein gerichtete Graph G mit ausgezeichneten Knoten v_{Start} und v_{Ziel} gegeben. Die Aufgabe besteht darin, einen Pfad, also eine Folge von aufeinanderfolgenden Kanten, zu berechnen, der in v_{Start} seinen Anfang hat und in v_{Ziel} endet. Dabei muss jeder Knoten genau einmal besucht werden. Abbildung 7.2 zeigt das von Adleman auf molekularer Ebene gelöste Problem.

$(0 \rightarrow 1)$	Path(edge(0,1)), -Visited(0), -Visited(1)
$(0 \rightarrow 6)$	Path(edge(0,6)), -Visited(0), -Visited(6)
	$\ddots$
$(5 \rightarrow 2)$	Path(edge(5,2)), -Visited(5), -Visited(2)
$(5 \rightarrow 6)$	Path(edge(5,6)), -Visited(5), -Visited(6)



(CONNECTOR)	-Path(edge(X,Y)), -Path(edge(Y,Z)), Path(edge(edge(X,Y),Z))
(SOLVER)	-Path(edge(edge(edge(edge(edge(edge(0,X2),X3),X4),X5),X6),6))
(TERMINATION)	-Visited(0), -Visited(1), -Visited(2), ... , -Visited(5), -Visited(6)

Abbildung 7.2: Repräsentation des Adleman-Problems in *RESAC*.

Um das Problem im Prädikatenkalkül erster Ordnung zu repräsentieren, müssen die im Graphen G kodierten Informationen in eine Folge von Klauseln in konjunktiver Normalform transformiert werden. Eine mögliche Umsetzung unter Zuhilfenahme der Funktion *edge()* und der Prädikate *Path()* und *Visited()* ist erneut in Abb. 7.2 tabelliert. Deutlich werden zwei Sektionen. Die erste Gruppe von Klauseln spiegelt die Graphspezifikation mit Knoten und Kanten wider. In der zweiten wird der (CONNECTOR) eingeführt, der die Tatsache ausdrückt, dass ein Pfad $X \rightsquigarrow Z$ existiert, wenn zwei Pfade $X \rightsquigarrow Y$ und $Y \rightsquigarrow Z$ gegeben sind. Die „lösende“ Klausel (SOLVER) komplettiert die Definition der Startklauseln, die die Eingabe des Systems *RESAC* ausmachen.

Eine vereinfachende Sicht des Lösungsprozesses ist dann wie folgt: Wo immer möglich verbindet der (CONNECTOR) kürzere Pfadstücke zu längeren Pfaden. Bei solchen Resolutionen akkumulieren sich die nicht aktiv am Lösungsprozess beteiligten *Visited()*-Literale. Ein Pfad $v_{Start} \rightsquigarrow v_{Ziel}$ mit 7 Knoten unifiziert und reagiert mit dem (SOLVER). Dabei wird das *Path()*-Literal resolviert und gleichzeitig die angesammelten *Visited()*-Literale freigesetzt. Das Terminierungskriterium (TERMINATION) spürt dann *Visited()*-Ansammlungen auf, die alle Knoten von G umfassen – und beendet *RESAC*. Wenn also ein Reaktionsprodukt dieser Klausel gleicht,[4] so ist eine Lösung dieses

[4] Es sei noch einmal erwähnt, dass das Terminierungskriterium keiner Resolutionsanwendung unterzogen wird, sondern auf Gleichheit mit einer fraglichen Klausel getestet wird.

Entscheidungsproblems gefunden, nämlich ein Pfad der Länge 6, der jeden Knoten besucht (folglich keinen Knoten zweimal). Außerdem wird durch den (SOLVER) aus Abb. 7.2 festgelegt, dass solch ein Pfad an Knoten 0 zu beginnen und an Knoten 6 aufzuhören hat. In den hier durchgeführten Experimenten gehört nur der (SOLVER) der Unterstützungsmenge (siehe Resolutionsstrategien) an; damit wird eine Art von rückwärtsgerichteter Suche (engl. backward chaining) induziert.

7.3 Experimente mit der DHPP-Chemie

Die mit *RESAC* durchgeführten Experimente arbeiten auf der gerade konstruierten Eingabeklauselmenge. Dabei werden verschiedene Algorithmen zur Steuerung der KC (Varianten von *A*) eingesetzt und verglichen: 6 Zufluss-Varianten und 4 Reaktorgrößen, jeweils unter den beiden Ersetzungsmethoden Eduktersetzung und freie Ersetzung. Jedes dieser Experimente umfasst dabei 100 Läufe. Abbildung 7.4 vergleicht die erzielten Erfolgsraten. Tabelle 7.2 untersucht, in welcher Generation im Durchschnitt eine Lösung gefunden wurde (durchschnittliche *Erfolgsgeneration* genannt) und listet die zugehörige Standardabweichung sowie den Standardfehler auf.

	Zuflussrate i	$\|M\| = 2000$	$\|M\| = 10000$			$\|M\| = 20000$			$\|M\| = 30000$		
freie Ersetzung	0	-	-	-	-	-	-	-	-	-	-
	0.01	-	-	-	-	**41.2**	*14.9*	1.49	**37.3**	*15.4*	1.54
	0.05	-	**40**	*16.5*	1.65	**30.3**	*11*	1.1	**26.4**	*8.4*	0.84
	0.1	-	**35.3**	*12.9*	1.29	**27.2**	*9.4*	0.94	**25.7**	*7*	0.7
	0.2	-	**39.8**	*17.5*	1.75	**30.4**	*10.3*	1.03	**26.1**	*7.8*	0.78
	elastisch	-	-	-	-	-	-	-	-	-	-
Eduktersetzung	0	-	-	-	-	-	-	-	-	-	-
	0.01	-	-	-	-	-	-	-	-	-	-
	0.05	-	-	-	-	**42.7**	*13.6*	1.36	**36.8**	*10.5*	1.05
	0.1	-	-	-	-	-	-	-	**39.1**	*12.8*	1.28
	0.2	-	-	-	-	-	-	-	-	-	-
	elastisch	-	-	-	-	-	-	-	-	-	-

Tabelle 7.2: Verschiedene Versuchsreihen zum Adleman-Experiment in *RESAC*: $|M|$ bezeichnet die Reaktorgröße. Jedes Experiment wurde 100 Mal wiederholt. Dabei wurde die Laufzeit auf maximal 100 Generationen beschränkt. Aufgelistet wird die durchschnittliche Erfolgsgeneration (**fett** gesetzt), die Standardabweichung (*kursiv* gesetzt) und der Standardfehler. „-“ zeigt an, dass nicht alle Experimente der entsprechenden Serie Erfolg zeigten.

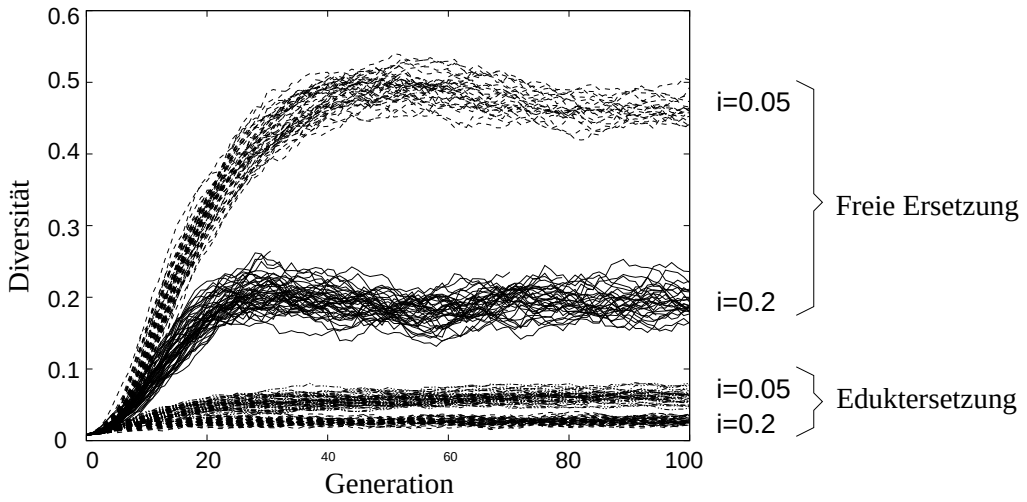


Abbildung 7.3: Diversität in Reaktoren der Größe 2000. Eduktersetzung kontra freie Ersetzung. Die Diversität ist signifikant höher bei der freien Ersetzung. *i* bezeichnet die Zuflussrate.[5]

Aus den gesammelten Daten können die folgenden drei Beobachtungen abgeleitet werden:

1. Das Schema der freien Ersetzung ist dem der Eduktersetzung in allen Experimentserien entweder überlegen oder aber ebenbürtig, und zwar unter dem Aspekt einer höheren Erfolgsrate, einer niedrigeren durchschnittlichen Erfolgsgeneration und einer niedrigeren Variabilität.
2. Es scheint eine besonders günstige, mit i_{opt} bezeichnete, Zuflussgeschwindigkeit zu geben, so dass für alle Zuflussraten $i > i_{opt}$ gilt, dass je kleiner i gewählt wird, desto größer die Erfolgsrate ist. Andererseits gilt aber für $i < i_{opt}$, dass sich eine umso niedrigere Erfolgsrate einstellt, je kleiner die Zuflussrate ist.
3. Mit zunehmender Reaktorgröße steigt die Erfolgsrate. Außerdem wird eine Lösung im Durchschnitt in früheren Generationen gefunden.

7.4 Analyse der experimentellen Resultate

In den durchgeführten Experimenten mit der DHPP-KC konnten in der Hauptsache die drei im vorangegangenen Kapitel beschriebenen Phänomene beobachtet werden. Diese Beobachtungen werden im Folgenden der Reihe nach analysiert.

7.4.1 Freie Ersetzung versus Eduktersetzung

Bei der Erklärung der ersten Beobachtung spielt die Diversität eine wichtige Rolle. Die Diversität der Moleküle in einem Reaktor mit freier Ersetzung ist signifikant höher als bei Anwendung der Eduktersetzung (Abb. 7.3 zeigt Beispiele dafür). Das hat eine bessere Verteilung der Lösungskandidaten (Klauseln) im Suchraum zur Folge.

[5] Eine zufällige Auswahl von 10 Läufen ist gleichzeitg dargestellt.

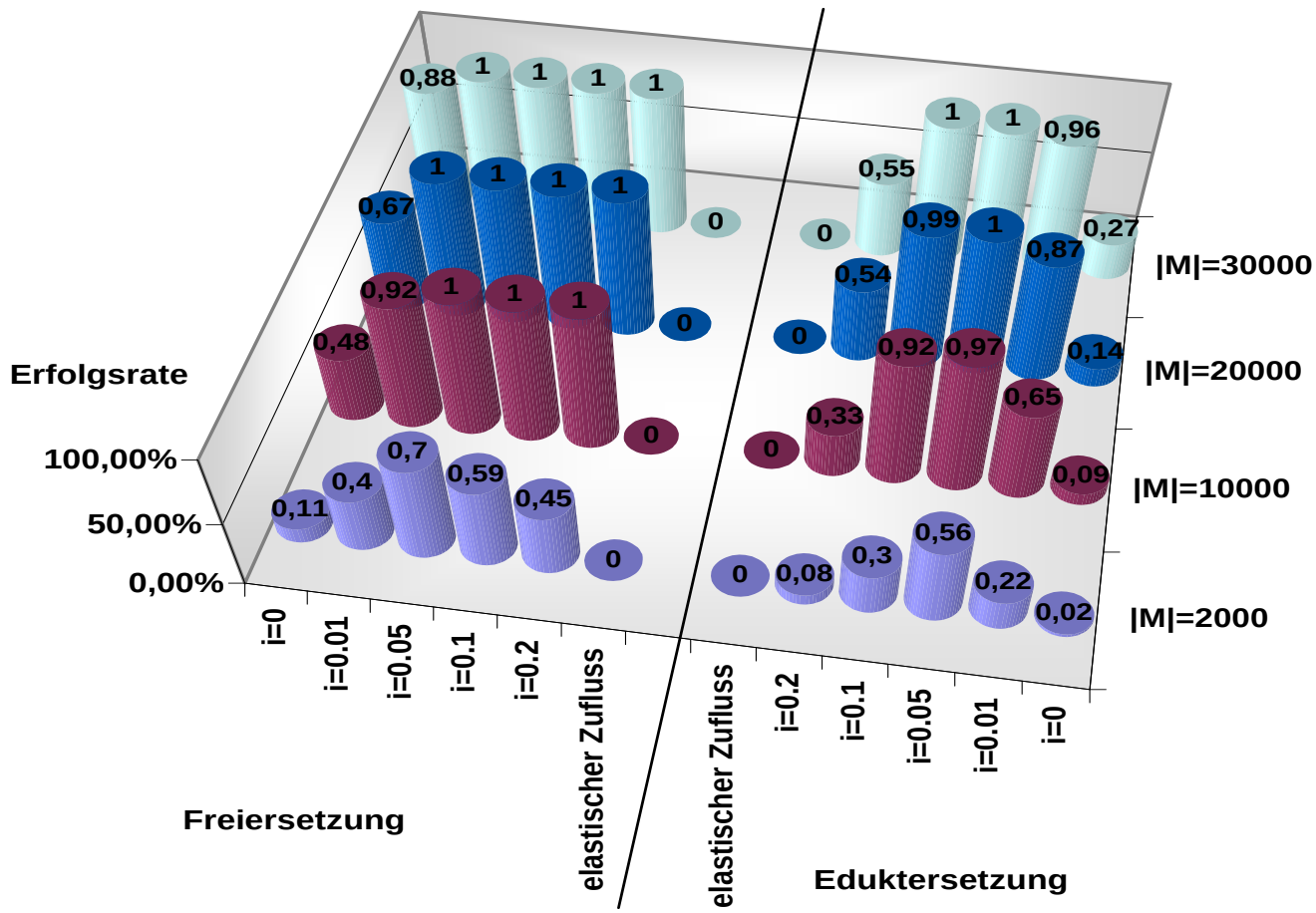


Abbildung 7.4: Das Adleman-Experiment in *RESAC*: Variiert wird die Ersetzungsmethode, der Zufluss und die Reaktorgröße. i bezeichnet die Zuflussrate, $|M|$ die Reaktorgröße. Jedes Experiment wurde 100 Mal wiederholt. Dabei wurde die Laufzeit auf maximal 100 Generationen beschränkt. Der Graph zeigt die Erfolgsraten der verschiedenen Experimente. Eine Zuflussrate von 5% führt zu den besten Resultaten.

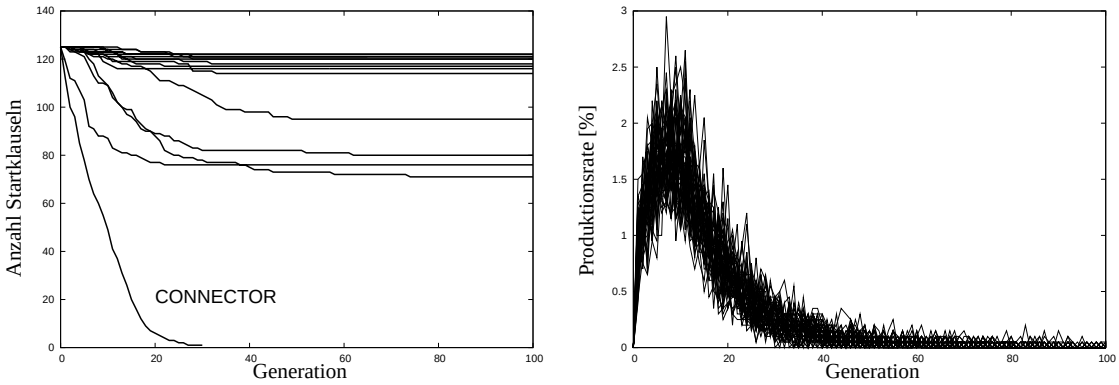


Abbildung 7.5: Analyse eines geschlossenen Reaktors der Größe 2000 basierend auf der Eduktersetzung. *Links:* Dynamik in der Zahl der Startklauseln. Beachtenswert die Konzentrationsentwicklung des CONNECTORS. *Rechts:* Produktivitätsraten in gleichen Reaktoren.[5]

Gründe dafür zu benennen erfordert eine tiefer gehende Betrachtung. Eine Konzentrationsverlaufsanalyse der Startklauseln in einem geschlossenen Reaktorsystem mit Eduktersetzung (Abb. 7.5) enthüllt das Aussterben des CONNECTORS zu einem relativ frühen Zeitpunkt des Experiments. Der Rückgang der Produktivitätsrate ist die Folge davon – sie nähert sich schließlich der Null-Prozent-Marke. Worin aber können die Ursachen für eine solch dezidiert erfolgende Verdrängung einer bestimmten Klausel oder Klauselgruppe liegen? Dieses Phänomen ist auch aus anderen Experimenten bekannt. Bei der Eduktersetzung wird genau eines der Edukte durch das Reaktionsprodukt ersetzt, eine produktive Reaktion vorausgesetzt. Mit anderen Worten: Bei der Eduktersetzung werden reaktive Moleküle gezielt, also schneller ersetzt. Unter diesem Aspekt könnte man demzufolge von einer „Bestrafung“ der an erfolgreichen Reaktionen teilnehmenden Moleküle sprechen. Konsequenz dieses Umstandes ist die Unfähigkeit, Reaktionswege dauerhaft aufrechtzuerhalten, sofern dem Reaktor kein Zufluss gewährt wird oder die einzelnen Bestandteile des Reaktionspfads nicht aus sich selbst heraus neugebildet werden können. Dieser Fall liegt mit der CONNECTOR-Klausel in der besprochenen Experimentserie mustergültig vor.

Darüber hinaus bleibt der totale Verlust einer Molekülsorte nicht ohne Folgen auf andere, quervernetzte Reaktionspfade, die ebenfalls von diesem Molekül abhängen. Andererseits sind nichtreaktive Moleküle offensichtlich nur an elastischen Kollisionen beteiligt und behalten unter der Eduktersetzung ihre Konzentration unverändert bei (Nischenbildung). Vermutlich unnütz in diesen bestimmten Zeitfenstern des Experiments (im Zuge der dynamischen Veränderung solcher Systeme kann sich die Rolle ändern und vielfältig, komplex sein), verbrauchen sie dennoch (wertvollen) Platz im Reaktor. Wie schon bei der theoretischen Betrachtung der Ersetzungsmethoden in Kap. 4 an einer Basisreaktion gezeigt, wird auch in diesem Experiment deutlich, dass das freie Ersetzungsschema einen *globalen Einfluss* hat und so einen globalen Druck ausübt.

Letztlich sei noch angemerkt, dass freie Ersetzung auch zu besseren Resultaten führt, selbst wenn der CONNECTOR nicht völlig ausstirbt, z. B. in offenen Reaktoren mit Eduktersetzung unter Zufluss-Variationen. Wiederum eröffnet die höhere Konzentration dieser Klausel mehr und vielfältigere Reaktionswege, was sich in einer klar höheren Diversität des Reaktors im Vergleich zur Eduktersetzung niederschlägt (siehe erneut Abb. 7.3).

7.4.2 Optimale Zuflussrate

Die zweite Beobachtung lässt eine besonders günstige Zuflussrate vermuten. Eine Abweichung von dieser Rate in beide Richtungen führt in den durchgeführten Experimenten zu Verschlechterungen bei den Erfolgsraten. Je höher der Zufluss ist, desto höher ist der Anteil der Startklauseln an der Gesamtpopulation (Abb. 7.6); sie blockieren das Reaktionssystem durch ihren Platzverbrauch, ohne dabei eine ausreichende Informationsvielfalt zu gewähren. Sehr deutlich demonstriert diese Experimententwicklung auch der elastische Zufluss (Abb. 7.7). Eine kleinere Zuflussrate dagegen verursacht eine höhere Diversität und führt damit zu einer besseren Abdeckung des Suchraums.

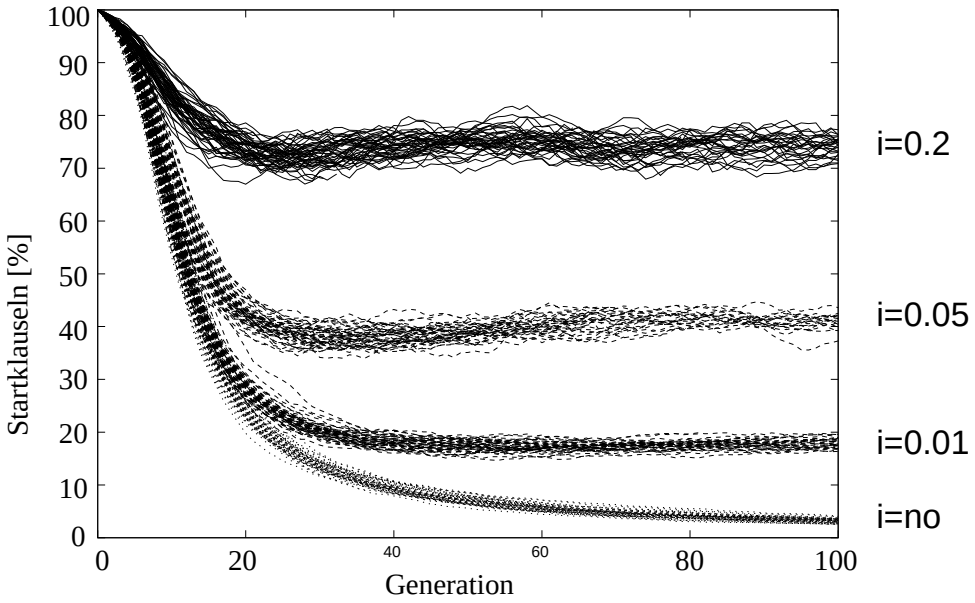


Abbildung 7.6: Vergleich von Zufluss-Varianten i in einem Reaktor der Größe 2000 basierend auf freier Ersetzung.[5]

Allerdings ist die Zuflussrate aber auch nicht zu klein anzusetzen, denn das erhöht die Wahrscheinlichkeit des Aussterbens einer gegebenenfalls für den Lösungsprozess relevanten Startklausel. Selbst wenn eine relevante Molekülsorte nicht ganz verschwindet, ein starker Konzentrationsabfall ist für eine Verlängerung der Lösungssuche verantwortlich, da Reaktionen infolge von Kollisionen zu einem seltenen Ereignis werden. Die günstigste Zuflussrate i_{opt} liegt also zwischen den Extremwerten, für den allgemeinen Fall ist sie schwer zu bestimmen (Problemabhägigkeit). In den hier durchgeführten Experimenten hat sich ein 5%-Zufluss als optimal erwiesen.

7.4.3 Reaktorgröße

Einen wichtigen Einfluss auf den Experimentverlauf hat die Reaktorgröße wie Abb. 7.4 auch zeigt. Durch Vergrößerung des Reaktorvolumens (was im vorliegenden Versuchsaufbau einer Vergrößerung der Population entspricht) kann eine niedrige Diversität kompensiert werden, die zum Beispiel ihre Ursache in einer hohen Zuflussrate hat. Dieser Sachverhalt wird deutlich an folgendem Vergleich: In der 20. Generation besteht ein Reaktor mit freier Ersetzung mit 20000 Molekülen

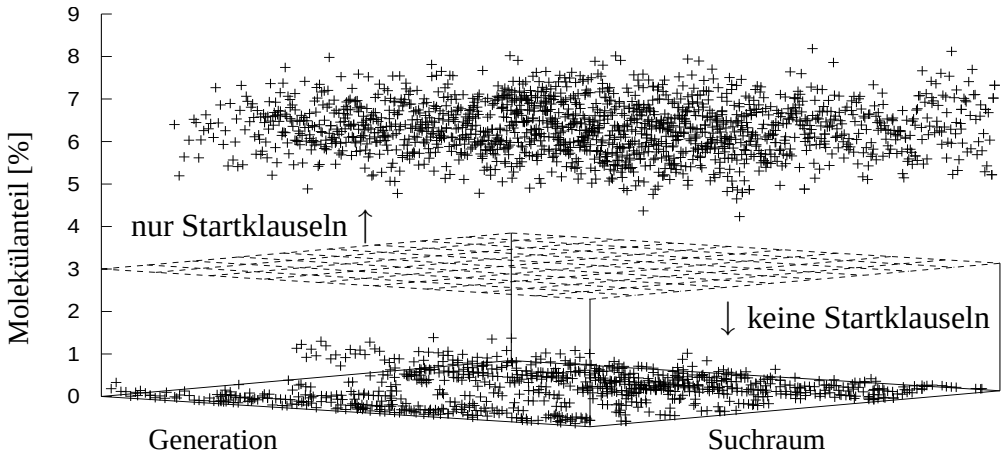


Abbildung 7.7: Elastischer Zufluss in einem Reaktor der Größe 2000 (freie Ersetzung): Der Graph zeigt alle vom System über die Zeit generierten Klauseln. Sie sind abgebildet auf den Integer-Zahlenbereich, der so den Suchraum darstellt. Alle Startklauseln haben hohe Konzentrationen (über 3%). Sie blockieren so Reaktorplatz und als Folge davon das Experiment. Alle erzeugten Nicht-Startklauseln haben Konzentrationen kleiner 2%.

und 20 prozentigem Zufluss im Durchschnitt aus ca. 2000 verschiedenen Molekülen (entspricht einer Diversität von 10%).[6] Das sind etwa dreimal mehr verschiedene Moleküle als ein kleinerer Reaktor der Größe 2000 mit 5 prozentigem Zufluss und größerer Diversität produziert (siehe Abb. 7.3, freie Ersetzung).

Im Allgemeinen werden in großen Reaktoren mehr Moleküle neu gebildet oder wiederholt gebildet. Abbildung 7.8 vergleicht Reaktionsnetzwerke zweier verschieden großer Reaktoren und macht Unterschiede in der Vielfalt der Population deutlich. Durch diesen Umstand – also der verbesserten Exploration des Suchraums – ist es in großen Reaktoren wahrscheinlicher eine Lösung in einer frühen Phase des Experiments zu finden als in kleineren. Dies belegt die dritte aus dieser Versuchsreihe abgeleiteten Beobachtung.

Wenn man sich noch einmal die Definition der Generation (Unterkapitel 2.6) vergegenwärtigt, wird jedoch deutlich, dass eine kleinere Erfolgsgeneration aber keineswegs auf eine schnellere Problemlösung in der Simulation einer Künstlichen Chemie rückschließen lässt. Wird die Ausführungszeit in Reaktionen gemessen, so führt der oben konstruierte große Reaktor 10 Mal mehr Reaktionen durch als seine kleinere Ausgabe. Ein Lauf verbraucht also, den konventionellen Begriff des Zeitverbrauchs (engl. computational time) zugrunde gelegt, mehr Rechenzeit. Die letzte Aussage gilt jedoch nur in Systemen, in denen die Reaktionen nicht parallel durchgeführt werden. In parallel arbeitenden Systemen bieten große Reaktoren im Durchschnitt tatsächlich Laufzeitvorteile.

[6] Werte sind empirisch aus einer separat durchgeführten Versuchsreihe entnommen.

7.5 Zusammenfassung

Dieses Kapitel hat gezeigt, wie ein Entscheidungsproblem in einer resolutionsbasierten Künstlichen Chemie implementiert werden kann. Dabei ist ein Perspektivwechsel vom automatischen Theorembeweiser *RESAC* zum Programmiersystem *RESAC* motiviert und vollzogen worden. Die Anreicherung einer Künstlichen Chemie durch Konzepte der Logik ermöglicht eine schnelle Anpassung des allgemeinen Konzepts an konkrete Problemanforderungen. Möglich wird das durch die dauerhafte Festlegung der Objektsyntax und der Reaktionsregeln. Was bleibt ist die Übertragung des Problems in die Sprache der Logik, hier der Prädikatenlogik 1. Ordnung. Es handelt sich daher mehr um eine deskriptive als um eine imperative Problembeschreibung. Im Unterschied zur logischen Programmierung wird bei der resolutionsbasierten Programmierung in *RESAC* aber nicht die Semantik der Logik ausgenutzt. Allein die Struktur der Sprache zusammen mit dem Konzept einer Künstlichen Chemie und der Reaktionsregel Resolution liefert die Problemlösung in einem parallelen, auf Emergenz beruhenden Prozess.

Der Skizze nötiger Konvertierungsschritte von der Problemspezifikation zur „Eingabe“ der Künstlichen Chemie folgte eine beispielhafte Programmierung des bekannten Adleman-Problems. Dieses Entscheidungsproblem wurde in die Syntax der Prädikatenlogik 1. Ordnung übertragen und eine Versuchsreihe in der *RESAC*-Simulation gestartet. 48 Varianten wurden unter den Gesichtspunkten der Performanz und des Dynamikverlaufes analysiert.

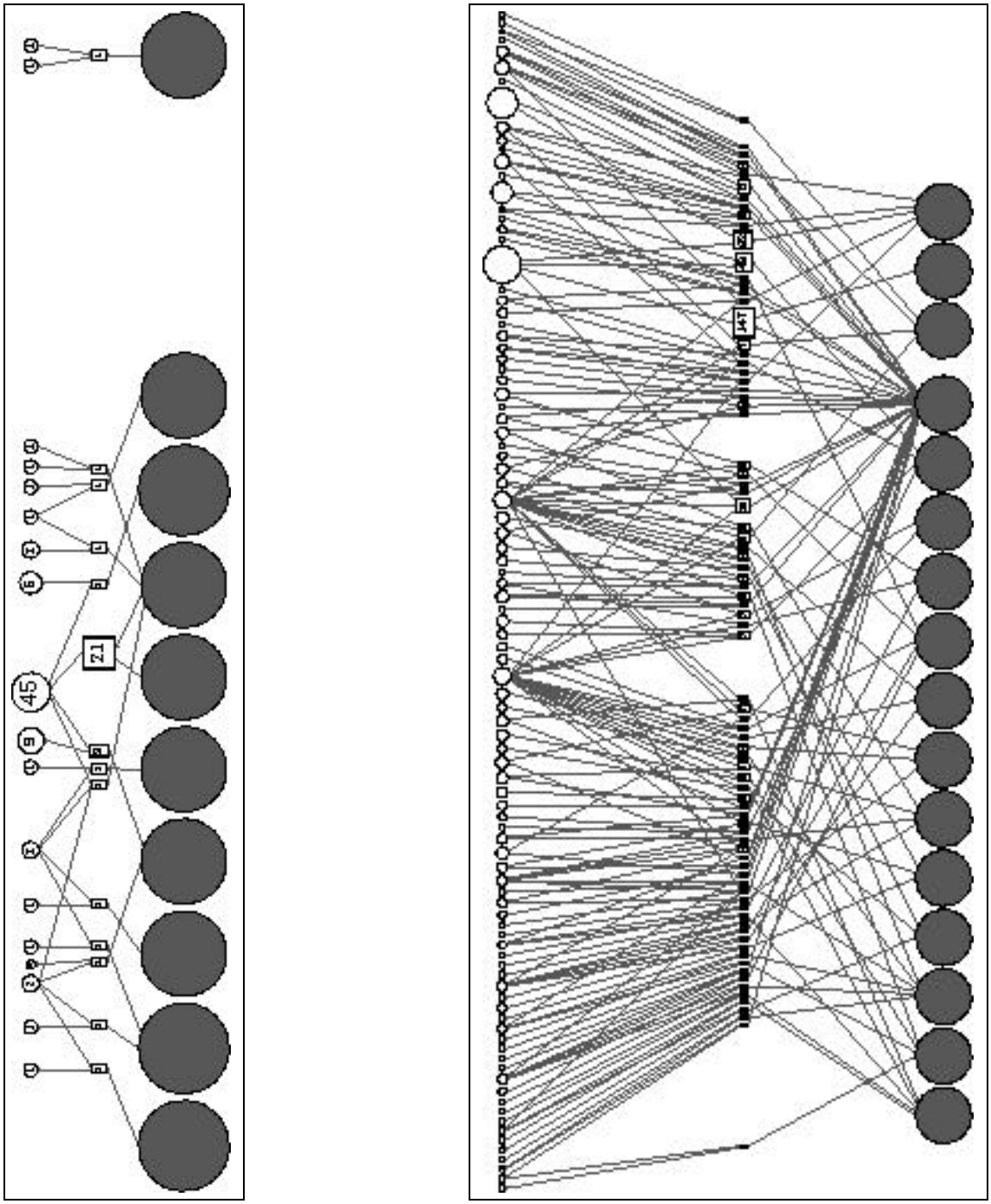


Abbildung 7.8: Reaktionsnetzwerk aus der 4. Generation, freie Ersetzung und 5%-Zufluss. Die Gegenüberstellung von Reaktionsnetzwerken soll nur einen Eindruck der unterschiedlichen Komplexität zweier Reaktoren vermitteln. Details sind bei dieser Betrachtung ohne Bedeutung. Jedes Rechteck der mittleren Reihen repräsentiert eine Reaktion. Die Linien zeigen die an der Reaktion beteiligten Edukte und das Produkt an. Kreise zeigen Moleküle an, dabei stehen die Kreise der jeweils rechten Reihen für Startklauseln. *Links:* Reaktor mit 2000 Molekülen. *Rechts:* Reaktor mit 20000 Molekülen. Offensichtlich werden hier sehr viel mehr Reaktionswege und neue Moleküle gebildet.

Kapitel 8

Lösung von Optimierungsproblemen mit *RESAC*

Dieses Kapitel wird Aufschluss darüber geben, wie Optimierungsprobleme in der resolutionsbasierten Künstlichen Chemie *RESAC* formuliert und gelöst werden können. Dabei wird in Fortführung des in Kapitel 3 eingeführten Basissystems ein allgemeiner Lösungsansatz angestrebt. Spezielle, problemspezifische und komplexe Optimierungsmethoden, die die Kollisionsbehandlung mit gegebenenfalls nachfolgender (chemischer) Reaktion mit zusätzlicher Komplexität befrachten, vermutlich überfrachten, sind nach heutigem Kenntnisstand eher ungeeignet für einen Einsatz in natürlichen, auf massiver paralleler Interaktion beruhenden Systemen. Solche Systeme mit einer ungeheuren Vielzahl von parallelen Berechnungen mit lokalem Wissenshintergrund sind weitgehend unverstanden und oft durch elementare Konzepte modelliert.

Das Ziel ist die Entwicklung eines Zusatzmechanismus, der für im Prädikatenkalkül formulierte Probleme ein allgemeines Konzept bereitstellt, das es ermöglicht, auf den angesprochenen Systemen eine Optimierung durchzuführen. Das System *RESAC* produziert durch parallele lokale Interaktion eine Vielzahl von möglichen Lösungen. Auf in diesem Kapitel beschriebene Weise können optimale Ergebnisse aus diesen selektiert werden.

Diese nicht-konventionelle Art der Optimierung fußt, wie erwähnt, auf parallel durchgeführten Massenkalkulationen und stellt eine Übertragung der durch Adleman (1994) ins Leben gerufenen Idee zum molekularen Rechnen (engl. molecular computing) dar. Effizient wird dieser Ansatz durch die Realisierung in einem Umfeld, das diese hohe Parallelität von Natur aus tatsächlich bietet. Als konkurrenzfähig und möglicherweise überlegen wird sich die vorgestellte Methode vermutlich etwa im Bereich des molekularen Rechnens zeigen. Eine Umsetzung liegt derzeit nicht vor. Daher beruhen die vorgeführten Experimente auf Software-Simulationen, die große Reaktoren modellieren. Die Umsetzbarkeit in reale (d. h. nicht-simulierte) Umgebungen soll aber in der Anlage des Optimierungsprozesses immer eine besondere Rolle spielen und Beachtung finden.

Es gibt grundsätzlich verschiedene Herangehensweisen, eine Optimierung in einer Künstlichen Chemie durchzuführen. Eine Möglichkeit ist die Veränderung von Molekülen durch Reaktion genau dann, wenn eine im Sinne des Optimierungskriteriums günstigere Lösung konstruiert wird. Das setzt jedoch problemspezifisches Wissen und oft auch eine komplexe Reaktionsdurchführung voraus. Eine zweite Möglichkeit ist die Entkopplung der eigentlichen Problemlösung von der Op-

timierung. Mit fortschreitender Experimentalzeit sammeln sich bei diesem Vorgehen Moleküle im Reaktor an, die eine Lösung repräsentieren. Diese müssen in einem Nachverarbeitungsschritt einzeln evaluiert und anschließend verglichen werden. Denkt man an eine Realisierung der Berechnung in einem natürlichen System, wird klar, dass das mit sehr hohem Aufwand verbunden ist. Wie sollten beispielsweise beim DNA-Computing 10^{15} Moleküle einzeln aufbereitet werden? Auch wäre das wiederum ein problemspezifischer Prozess.

Die dritte Möglichkeit besteht darin, schon während der Problemlösung die Kosten bzw. den Nutzen zu protokollieren. In jedem einzelnen Molekül liegen dann, wenn eine Lösung gefunden wurde, auch die Kosten dieser Lösung vor. Eine nachgeschaltete Evaluierung entfällt. Wenn es möglich wäre, diese Kosten strukturell auszudrücken, könnte das Ausfiltern der günstigsten Lösung auf Struktursondierungen herauslaufen. Solche Sondierungen sind meist weniger komplex, beispielsweise eine Auftrennung der Moleküle nach Länge oder Gewicht.

Dieses Kapitel zeigt eine „Implementierung“ des dritten Verfahrens. In Unterkapitel 8.2 wird eine reaktionsbegleitende Kostenzählung eingeführt. Voraussetzung dafür ist, dass in irgendeiner Weise addiert werden kann. Das folgt direkt aus dem Ziel, ein Attribut des Systems im Sinne der Aufgabenstellung zu optimieren. Die Frage, wie in einem resolutionsbasierten System wie *RESAC* eine für diesen Zweck dienliche Addition bewerkstelligt werden kann, ist daher von grundlegender Bedeutung für Aufgabenstellungen aus dem Bereich der Optimierung. Eine Antwort gibt das nächste Unterkapitel. Darin wird spezielle Aufmerksamkeit auf ein Verfahren der strukturellen Addition verwendet. Dieses Verfahren setzt den Summationsprozess direkt in eine Veränderung der Molekülstruktur um. Eine solche strukturgebende Addition ist nicht auf triviale Weise in einem resolutionsbasierten System zu verwirklichen und die Herleitung bzw. der Korrektheitsbeweis etwas mühsam. Aber der Nutzen ist möglicherweise hoch, wie das erwähnte Verfahren der Gel-Elektrophorese zeigt, die DNA-Moleküle der Größe nach auftrennt.

8.1 Resolutionsbasierte Addition

Dieses Kapitel beschäftigt sich mit der Addition über dem Zahlenbereich der natürlichen Zahlen $\mathbb{N}$. Der Aufbau einer Form von Arithmetik zeigt einerseits die Ausdrucksstärke der Prädikatenlogik 1. Ordnung (siehe auch Kap. 1.4), andererseits ist die axiomatische Beschreibung einer irgendwie gearteten Arithmetik der Grundstein für die Formalisierung bzw. den Ausdruck mathematischer Fakten. Zusammenhänge aus dem Bereich der Algebra können innerhalb einer solchen Arithmetik formuliert und Beweisführungen angetreten werden. Tatsächlich ist dies nach Genesereth und Nilsson (1989) die ursprüngliche Motivation, die zur Entwicklung des Prädikatenkalküls führte. An diesem Anspruch hat sich bis heute nichts verändert.

Anstatt sich nun mit einer ganzen Arithmetik zu beschäftigen, genügt es – wie oben angesprochen – im Rahmen der Lösung von Optimierungsaufgaben, sich auf die Addition zu beschränken. Es gibt mehrere Möglichkeiten, eine für diese Zwecke geeignete Addition in der Prädikatenlogik 1. Ordnung zu definieren. Zwei Möglichkeiten, die die resolutionsbasierte Summenbildung unterstützen, werden im Folgenden vorgestellt.

8.1.1 Semantische Addition

Den Überlegungen vorangestellt wird eine Auflistung der Additionstheoreme der *Peano-Arithmetik 1. Ordnung* (siehe auch Kapitel 1.4):

$$\forall m\ NaturalNumber(m) \Rightarrow +(m,0) = m.$$

$$\forall m,n\ NaturalNumber(m) \wedge NaturalNumber(n) \Rightarrow +(s(m),n) = s(+(m,n)).$$

Diese Axiome begründen und formalisieren die Addition auf Basis der natürlichen Zahlen (hier durch das Prädikat *NaturalNumber* ausgedrückt) und einer Nachfolgefunktion s in der Prädikatenlogik 1. Ordnung. Eine Anleitung zur klauselbasierten Addition zweier Zahlen liefern sie jedoch nicht explizit. Eine mögliche operationalisierte Form, beispielsweise für „+5 “, ist durch folgende *Zahlklausel* gegeben:

$$\overline{P(X)}, P(s(s(s(s(s(X)))))).$$

Dabei erfüllt jedes der beiden Literale eine bestimmte Funktion: Das hintere gibt durch die Zahl der verschachtelten Nachfolgefunktionsaufrufe die darzustellende Zahl an, das vordere dagegen ist der reaktive Anteil, der durch Resolution eine Verbindung mit anderen Zahlklauseln ermöglicht. Die Summe $5+3$ ergibt sich etwa durch Resolution der Klauseln

$$\overline{P(X)}, P(s(s(s(s(s(X)))))$$

und

$$\overline{P(X')}, P(s(s(s(X'))))$$

zu

$$\overline{P(X'')}, P(s(s(s(s(s(s(s(X''))))))))).$$

Definition 8.1.1 (Semantische Zahlklausel)

Die *semantische Zahlklausel* $Z(P,n)$ zur Zahl n und gegebenem Prädikat P ist für $n > 0$ definiert als die Klausel $(l_1 \vee l_2)$ mit $l_1 = \overline{P(X)}$ und $l_2 = P(s^n(X))$. Das Literal l_2 wird *Zahlliteral* genannt. Für $n = 0$ ist $Z(P,n) = \overline{P(X)} \vee P(X)$.

Wird das Standardmodell der Peano-Arithmetik herangezogen, welches als Domäne die Menge der natürlichen Zahlen annimmt, und welches die Konstante 0 als Null interpretiert sowie die Nachfolgefunktion s als die Funktion, die die Zahl n auf $n+1$ abbildet, so ist durch das Zahlliteral von $Z(P,n), n > 0$, tatsächlich die Semantik der arithmetischen Zahl n gegeben. Im Gegensatz dazu aber entspricht das Zahlliteral $P(X)$ von $Z(P,0)$ nicht der Semantik der Null. Semantisch korrekt dagegen wäre das Zahlliteral $P(0)$. Dieses führte allerdings zu Fehlern in einer Kette von Additionen. Man betrachte dazu die Summe $y+0$, also die Resolution von

$$\overline{P(X)}, P(s^y(X)) \text{ und } \overline{P(X)}, P(0).$$

Ergebnis ist entweder

$$\overline{P(X)}, P(0) \text{ oder } \overline{P(X)}, P(s^y(0)).$$

In beiden Fällen gibt es keine reaktive Kopplung mehr zwischen den jeweiligen Literalen. Den Nachteil dieser semantischen Ungenauigkeit in Kauf nehmend übernimmt die Variable X in Def. 8.1.1 diese Aufgabe.

8.1.2 Strukturelle Addition

Die zweite Realisierung einer resolutionsbasierten Addition beruht auf einem anderen, syntaktischen Konstrukt. Hierbei wird die Zahl ausschließlich durch die Struktur kodiert, und zwar in Abhängigkeit von der Zahl der Literale in einer Klausel:

Definition 8.1.2 (Strukturelle Zahlklausel)
Die *strukturelle Zahlklausel* $W(P,n)$ zur Zahl n und gegebenem Prädikat P ist für $n > 0$ definiert als die Klausel, die aus Literalen

$$\text{(Anfangsglied)} \quad \overline{P(X,Y_0)}, \tag{8.1}$$

$$\text{(Zahlliterale)} \quad \overline{P(Y_{i-1},Y_i)} \quad \text{für } 1 \leq i \leq n, \tag{8.2}$$

$$\text{(Klammer)} \quad P(X,Y_n) \tag{8.3}$$

besteht. Für $n = 0$ ist $W(P,n) = \overline{P(X,Y)}, P(X,Y)$.

Nach dieser Definition ist die Anzahl der Literale in einer strukturellen Zahlklausel zur Zahl $n > 0$ proportional zu n. Neben den n Literalen aus (8.2) wird die Zahlklausel durch spezielle Literale an beiden Enden geklammert. Da alle Literale bis auf die Polarität von gleicher Grundstruktur sind, gibt es keinen ausgezeichneten reaktiven Teil wie bei der semantischen Zahlklausel.

In Abb. 8.1 ist $W(Add,4)$ dargestellt. Durch die Verzahnung der Variablen der zweistelligen Prädikate entsteht eine virtuelle Nachbarschaftsbeziehung auf den Literalen, die es unabhängig von der tatsächlichen Anordnung der Literale o. B. d. A. erlaubt, diese als Kette zu interpretieren.[1] Wichtig ist, dass durch das Klammerliteral Anfangsglied und letztes Zahlliteral zusammengefasst sind.

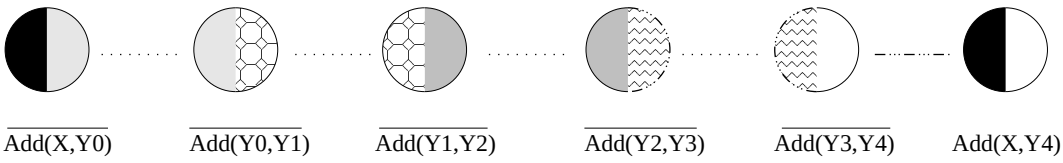


Abbildung 8.1: Schema der Zahlklausel $W(Add,4)$. Die Verbindung der Literale durch die Wahl der Variablen ist durch die Kettendarstellung symbolisiert. Die Literale sind verbunden durch die Logik, es besteht keine direkte physikalische Verbindung. Im einzigen Literal positiver Polung, der *Klammer*, steckt die Information über Anfang und Ende der Kette der negativen Literale.

Satz 8.1.1 (Resolution struktureller Zahlklauseln)
Die Resolution zweier struktureller Zahlklauseln $W(P,n)$ und $W(P,m)$ liefert wiederum eine Zahlklausel, und zwar genau $W(P,n+m)$.

Beweis: Die Fälle $n = 0$ und/oder $m = 0$ sind trivial. Sei nun daher $n,m \neq 0$. Die entscheidende Beobachtung ist, dass immer genau über die Klammer *einer* der beteiligten Klauseln resolviert

[1] Eine solche Kette denkt man sich dann in der natürlichen Ausrichtung Anfangsglied → Klammerliteral, so dass Begriffe wie „Anfang der Kette“, „linksseitig“ o. Ä. sinnig sind.

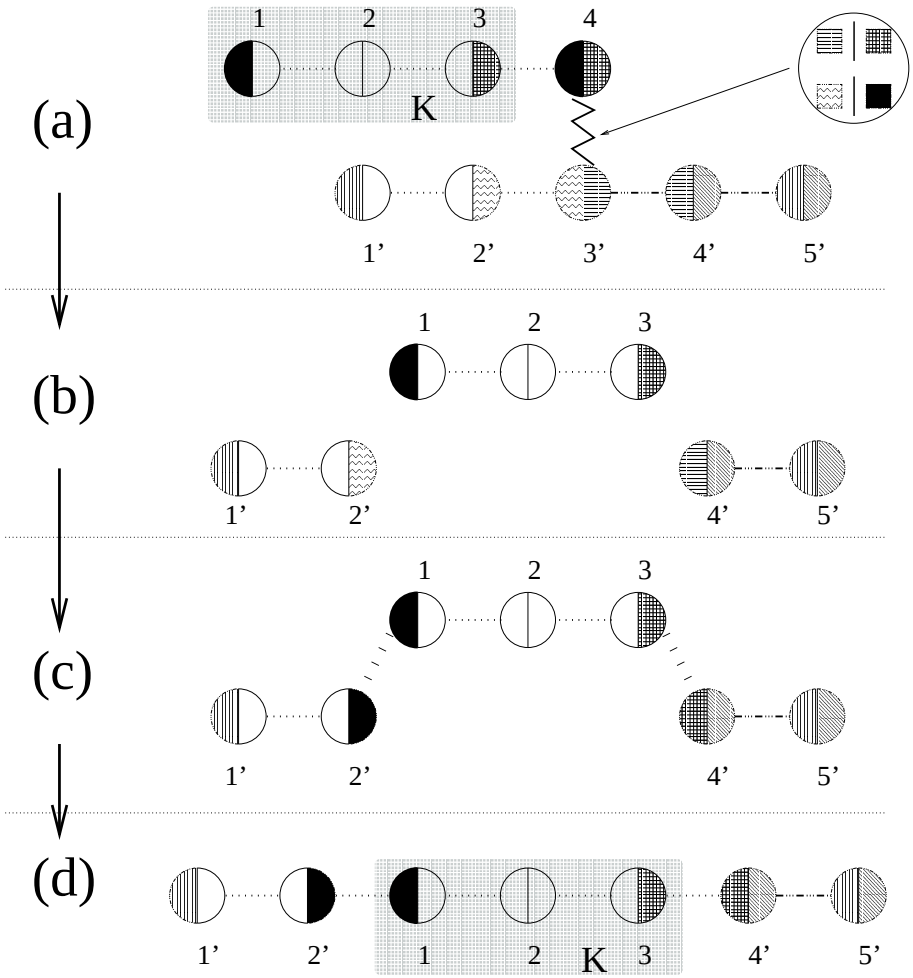


Abbildung 8.2: Schematische Darstellung des Resolutionsprozesses zwischen strukturellen Zahlklauseln. (a) zeigt die Anlagerung der Resolutionsliterale mit anschließender Unifikation, (b) das Wegschneiden der resolvierten Literale, (c) die Substitution der Variablen an den Schnittenden, (d) die Einpassung der Kette K.

wird, da eine Resolution immer nur zwischen Literalen entgegengesetzter Polarität stattfinden kann. Das Klammerliteral ist jedoch das einzige positive Literal einer Zahlklausel. Sei W_1 die Zahlklausel, deren Klammer resolviert wurde, und bezeichne weiterhin W_2 die andere.

Bei der Resolution unifizieren die Variablen des Klammerliterals von W_1 mit den Variablen eines negativen Literals aus W_2, welches im Folgenden *Einpassstelle* genannt wird. (Abb. 8.2 zeigt diesen Prozess). Dabei wird Information über Anfang und Ende der aus Anfangsglied und Zahlliteralen gebildeten Kette K von W_1 auf W_2 übertragen. Es sind nun 2 Fälle zu unterscheiden: Entweder ist die Einpassstelle das Anfangsglied von W_2 oder ein Zahlliteral. Im letzten Fall geschieht dieser Informationsübertrag nach Konstruktion durch die Substitution der zweiten Variable des zu der Einpassstelle linksseitig benachbarten Literals sowie der ersten Variablen des rechtsseitig benachbarten Literals. Diese beiden Variablen aus W_2 werden dadurch auf Anfang und Ende

von K gepolt. Nach „Absprengen“ der Einpassstelle und der Klammer passt sich K so genau in W_2 ein.

Ist aber die Einpassstelle Anfangsglied von W_2, so wird die erste Variable sowohl des ersten Zahlliterals als auch der Klammer entsprechend substituiert. In diesem Fall bildet K den Anfang der resultierenden Zahlklausel.

Nach diesen Vorbetrachtungen ist klar, dass immer die Kette, deren Endglied resolviert wurde, in die andere Kette eingesetzt wird. Es folgt ein formaler Beweis:

Es bezeichne

- $W_1.A_l, W_1.A_r$ die erste („linksstehende“) bzw. zweite („rechtsstehende“) Variable des Anfangsgliedes von W_1,
- $W_1.Z_l, W_1.Z_r$ die erste bzw. zweite Variable des letzten Zahlliterals von W_1,
- $W_2.A_l, W_2.A_r$ die erste bzw. zweite Variable des Anfangsgliedes von W_2,
- $W_2.E_l, W_2.E_r$ die erste bzw. zweite Variable der Einpassstelle von W_2,
- $W_2.Z_l, W_2.Z_r$ die erste bzw. zweite Variable des letzten Zahlliterals von W_2.

Zu zeigen ist, dass das Resultat einer Resolution zwischen W_1 und W_2 eine der Def. 8.1.2 genügende Struktur hat (i) und $n+m$ Zahlliterale aufweist (ii).

(i) Nach Wahl der Bezeichner findet die Resolution zwischen der Klammer von W_1 mit Variablen $W_1.A_l, W_1.Z_r$ und der Einpassstelle von W_2 mit Variablen $W_2.E_l, W_2.E_r$ statt. Es müssen also $W_1.A_l$ mit $W_2.E_l$ sowie $W_1.Z_r$ mit $W_2.E_r$ unifiziert werden. Das liefert in Abhängigkeit der Unifikation die Substitution $\sigma = \{W_2.E_l/W_1.A_l, W_2.E_r/W_1.Z_r\}$ oder die Substitution $\tau = \{W_1.A_l/W_2.E_l, W_1.Z_r/W_2.E_r\}$ oder $\gamma = \{W_1.A_l/W_2.E_l, W_2.E_r/W_1.Z_r\}$ oder $\delta = \{W_2.E_l/W_1.A_l, W_1.Z_r/W_2.E_r\}$. Im Folgenden wird nur die Unifikation betrachtet, die zur Substitution σ führt, die anderen Fälle sind analog zu beweisen.

   Die Einpassstelle ist nicht Anfangsglied von W_2: Klausel W_1 verliert das Klammerliteral und wird zu W_1', bleibt aber ansonsten zusammenhängend. Klausel W_2 verliert die Einpassstelle und zerfällt dadurch in die 2 Teile W_{21}' und W_{22}'. Diese Teilstücke selbst bleiben wieder zusammenhängend. Daher gilt: Innerhalb von W_1', W_{21}', W_{22}' sind die Ketten intakt und die Eigenschaften aus Def. 8.1.2 bleiben erhalten. Zu zeigen bleibt nur noch, dass die Übergänge zwischen den Teilstücken passen.

   Man verifiziert leicht, dass sich gültige Übergänge bei Zusammensetzung der Teilstücke in der Reihenfolge W_{21}'–W_1'–W_{22}' ergeben. Und zwar auf die folgende Weise: Das Literal $P(W_2.A_l, W_2.A_r)$ bildet das Anfangsglied des Resultats. Den Abschluss der Kette W_{21}' bildet die zweite Variable des letzten Literals von W_{21}', $(W_2.E_l\sigma) = W_1.A_l$. Die erste Variable des ersten Literals von W_1', $(W_1.A_l\sigma) = W_1.A_l$, sichert den korrekten Übergang W_{21}'–W_1'. Das Ende von W_1' ist durch Variable $(W_1.Z_r\sigma) = W_1.Z_r$ gegeben. Passend markiert $(W_2.E_r\sigma) = W_1.Z_r$ den korrekten Übergang W_1'–W_{22}'. Das Klammerliteral des Resultats ist nach Konstruktion das letzte Literal von W_{22}', nämlich $P(W_2.A_l, W_2.Z_r)$ – und damit korrekt.

Kein anderer Übergang ist möglich, da das Teilstück W_1' (bis auf die durch die Substitution betroffenen Variablen, die zu der Betrachtung im vorhergehenden Abschnitt führen) variablenfremd zu den Teilstücken W_{21}' und W_{22}' ist (Folge der Resolutionsanwendung: $\rightarrow$ Standardisierung der Variablen).

Die Einpassstelle ist Anfangsglied von W_2: Der Beweis verläuft analog; es bezeichnet dann W_{21}' ein leeres Teilstück und o. B. d. A. ist $\sigma = \{W_2.A_l/W_1.A_l, W_2.A_r/W_1.Z_r\}$ (alternative Substitutionen analog). Die Kette des Resultats startet also mit Literal $P(W_1.A_l, W_1.A_r)$. Die Variablen des Klammerliterals $(W_2.A_l, W_2.Z_r)\sigma = (W_1.A_l, W_2.Z_r)$ referenzieren daher auch in diesem Fall korrekt Anfang und Ende der durch W_1'–W_{22}' gebildeten Kette.

(ii) Die Zahl der Zahlliterale des Resolutionsresultats ergibt sich durch die folgenden Gleichungen, wobei $|A|_Z$ die Anzahl der Zahlliterale in Klausel A bezeichnet:

$$\begin{aligned} |\text{Resolvent } (W_1,W_2)|_Z &= (|W_{21}'|_Z + |W_{22}'|_Z) + |W_1'|_Z \\ &\overset{(*)}{=} (|W_2|_Z - 1) + (|W_1|_Z + 1) \\ &= (m-1) + (n+1) = m+n. \end{aligned} \tag{8.4}$$

In Schritt (*) wird ausgenutzt, dass die Einpassstelle in W_2 wegfällt und die Klammer von W_1 zum Zahlliteral wird.

Damit wurde gezeigt, dass das Resultat einer Resolution zwischen $W(P,n)$ und $W(P,m)$ die strukturelle Zahlklausel $W(P,n+m)$ liefert. □

Bemerkung: Bei der Wahl des Resolutionsliterals in W_2 (also der Einpassstelle) besteht hier Wahlfreiheit.

Satz 8.1.1 zeigt, dass strukturelle Zahlklauseln in einem resolutionsbasierten System zur Addition genutzt werden können. Dabei besitzen diese Eigenschaften, die sie darüber hinaus in verallgemeinerter Form auch für andere Anwendungsgebiete interessant machen. Das nächste Unterkapitel zeigt den Einsatz der Additionsklauseln in der Künstlichen Chemie *RESAC*.

8.1.3 Additionsexperimente in *RESAC*

In den letzten beiden Kapiteln wurden zwei verschiedene Repräsentationen für die resolutionsbasierte Addition angegeben. Bei der Lösung von Optimierungsproblemen wird es sich später (Kapitel 8.3.2) als vorteilhaft erweisen, jede Zahl mit dem ihrem Zahlenwert entsprechenden Gewicht zu assoziieren. Die in einem Reaktor vorhandenen Klauseln können dann entsprechend ihres Gewichts aufgetrennt werden, wobei das jeweilige Gewicht proportional zur Anzahl der Literale einer Klausel sein soll. Die Gewichte unterschiedlicher Zahlen sollten zur eindeutigen Trennbarkeit nicht zu ähnlich sein. Betrachtet man semantische und strukturelle Zahlklauseln unter diesem Aspekt, so gelten für aufeinanderfolgende Zahlen diese Gewichtsdifferenzen

$$\Delta^{\textit{Sem. Zahlkl.}}_{\textit{Gewicht},(n+1)\leftrightarrow n} = 2-2 = 0, \quad \Delta^{\textit{Strukt. Zahlkl.}}_{\textit{Gewicht},(n+1)\leftrightarrow n} = ((n+1)+2)-(n+2) = 1. \tag{8.5}$$

Strukturelle Zahlklauseln führen zu einer besseren Trennbarkeit. Im Folgenden werden sie zur Implementierung der Addition verwendet.

In einem ersten Experiment soll eine resolutionsbasierte Addition mit einer in einer freien Künstlichen Chemie formulierten verglichen werden.

Vergleich einer resolutionsbasierten mit einer frei-formulierten Addition

Dittrich et al. (2001) analysierten eine sog. Zahlen-Additions-Chemie, in der die Moleküle natürliche Zahlen repräsentieren. Der Reaktionsmechanismus ist einfach die Anwendung des normalen Additionsoperators:

$$s_1 + s_2 \longrightarrow s_1 + s_2 + \mathit{reaction}(s_1, s_2). \tag{8.6}$$

Die Funktion *reaction* ist definiert als $\mathit{reaction}(s_1, s_2) = s_1 + s_2 \bmod 2^n$. Im dort angegebenen Beispiel beträgt $n = 32$, also die Länge einer normaler Integerrepräsentation. Wird der Reaktor nur mit Kopien der Zahl 1 gefüllt, so ergibt sich eine typische Dynamik: Nach einer kurzen Übergangsphase erscheint der Reaktorinhalt wie zufällig erzeugt, er ist zu 100 Prozent divers – Abb. 8.3 zeigt dies.

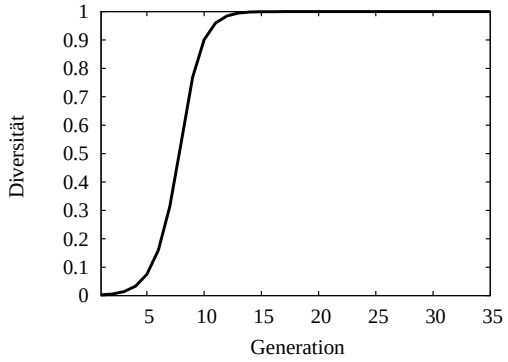
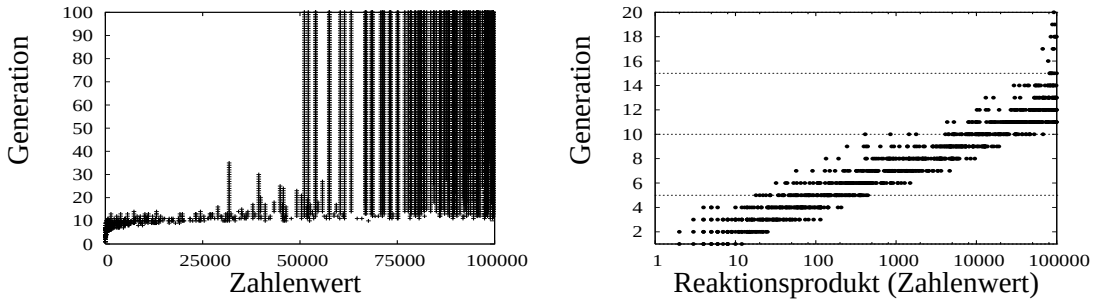


Abbildung 8.3: Simulation einer KC, die natürliche Zahlen addiert. Der Reaktor ist anfangs mit Molekülen befüllt worden, die ausschließlich Kopien der Zahl 1 sind. Sowohl in einer allgemeinen KC als auch in *RESAC* ergibt sich das gleiche Verhalten. Der Graph zeigt die relative Anzahl verschiedener Molekültypen in der Chemie über die Zeit. Nach ca. 15 Generation besteht der Reaktor annähernd nur noch aus verschiedenen Zahlen. Die Übergangsphase von einem anfangs homogen gefüllten Reaktor ist relativ kurz. Die Population umfasst hier 10000 Moleküle.

Wie erwartet zeigt *RESAC*, initialisiert mit Kopien der Zahlklausel $W(P, 1)$, das gleiche Verhalten und verifiziert so die obigen Resultate. Es soll noch einmal darauf hingewiesen werden, dass in der Künstlichen Chemie *RESAC* im Gegensatz zur KC von Dittrich et al. (2001) kein auf diese Anwendung zugeschnittenes Reaktionsschema verlangt wird. Das problemspezifische Moment liegt jedoch in der Struktur der Moleküle.

Hart-beschränkte, resolutionsbasierte Zahlen-Additions-Chemien

In der vorangegangenen Zahlen-Additions-Chemie wurden Zahlen, die einen vorgegebenen Wertebereich verließen, mittels Modulo-Rechnung wieder in den Anfang dieses Bereichs gespiegelt. Trotz eines beschränkten Wertebereichs ist daher die Addition gewissermaßen *durchlaufend* konstruiert. Dagegen soll von einer *hart-beschränkten* Additions-Chemie gesprochen werden, wenn



(a) (b)

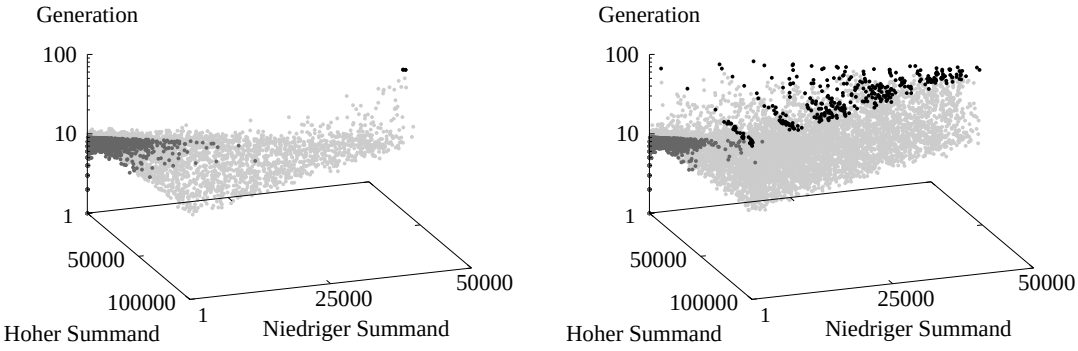
Abbildung 8.4: Hart-beschränkte Zahlen-Chemie in *RESAC*: Abgeschlossener Reaktor mit 100 Molekülen und Eduktersetzung, Zahlenschranke $Max = 100000$. Abb. (a) zeigt die pro Generation im Reaktor vorhandenen Zahlen. Schon nach kurzer Zeit sind nur noch Zahlen über 50000 vorhanden. Das exponentielle Wachstum der Zahlenwerte in den ersten 10 Generationen dokumentiert der Graph in (b). Der Zahlenwert der Summe einer jeden Addition (also das Reaktionsprodukt) ist hier logarithmisch aufgetragen. Im gezeigten Experiment liegt die Produktionsdauer einer Zahl zwischen 4 und 6 Generationen.

Kollisionen, die zu einem Überlauf führen, nicht zugelassen werden und so keine produktive Reaktion und damit Anwendung des Additionsoperators zur Folge haben. Von der Überlagerung der künstlich zurückgesetzten Zahlen entkoppelt, bieten solche Chemien besondere Einblicke in die zugrunde liegende Dynamik.

Zur Simulation hart-beschränkter Additions-Chemien werden mehrere Experimente mit dem System *RESAC* durchgeführt. Dabei ist eine Reaktion zwischen n und m nur dann produktiv, wenn $(n+m) \leq Max$ ist für eine beliebige, aber feste Zahlenschranke Max. Untersucht wurden geschlossene Reaktoren mit freier oder Eduktersetzung. Abb. 8.4 zeigt einige Eigenschaften der Dynamik dieser Systeme. Wie erwartet wachsen die Zahlenwerte im Reaktor exponentiell an, solange sie nicht durch die Begrenzung daran gehindert werden. Die Zahlenschranke sorgt dafür, dass nach einer gewissen Zeit fast[2] alle Zahlen im Reaktor in der oberen Hälfte des Zahlenbereichs liegen.

Dieser Zustand tritt bei Wahl der Eduktersetzung schnell auf. Bei jeder produktiven Reaktion wird eine kleinere Zahl durch eine größere ersetzt, es entsteht gleichsam eine Sogwirkung in Richtung Zahlenschranke. Es folgt außerdem, dass Zahlen, die so groß sind, dass sie durch weitere Addition nur größer als Max würden, niemals mehr ersetzt werden. Ist jedoch die freie Ersetzung implementiert, so können alle Zahlen ersetzt werden. Das vermindert den Druck auf Zahlen des unteren Zahlenbereichs. Abbildung 8.5 zeigt eine Auftrennung der an einer Reaktion beteiligten Summanden. Die Wahl des Ersetzungsschemas hat einen deutlichen Einfluss auf die Dynamik. Bei der Eduktersetzung ist es so, dass der „bearbeitete Problembereich“ (hier der Zahlenbereich) stark gekoppelt ist an die Zeit, denn im Verlauf des Experiments reagieren nur noch relativ große Zahlen miteinander (solange die Zahlenschranke das zulässt). Bei der freien Ersetzung sind auch in fortgeschrittenen Generationen Reaktionen zwischen kleinen Zahlen möglich. Wie schon bei anderen Experimenten konstatiert, arbeitet die Eduktersetzung lokal, die freie Ersetzung hingegen global.

[2] möglicherweise verbleibt eine einzige Zahl in der unteren Hälfte des Zahlenbereiches, nämlich dann, wenn es keinen Reaktionspartner aus der oberen mehr geben sollte, so dass die Summe kleiner als oder gleich Max wäre.



Eduktersetzung Freiersetzung

Abbildung 8.5: Hart-beschränkte Zahlen-Chemie in *RESAC*: Abgeschlossener Reaktor mit 1000 Molekülen, $Max = 100000$. Die Graphen zeigen eine Auftrennung der Summanden jeder Reaktion und machen damit das Einzugsgebiet produktiver Reaktionen zu gegebener Zeit sichtbar. Verschiedene Grauwerte heben drei Zeitabschnitte hervor: Mittelgraue Punkte kennzeichnen Reaktionen in den Generationen 1-9, schwach grau sind die Reaktionen der Generationen 10-79 eingefärbt. Schwarze Punkte zeigen die Reaktionen ab der 80. Generation. *Links:* Mit zunehmender Zeit finden Reaktionen nur noch mit hohen Summanden statt. *Rechts:* Bei freier Ersetzung ist das Anfangsverhalten ähnlich. Auch zu späten Zeitpunkten sind kleine Zahlen an Reaktionen beteiligt. Es zeigt sich das globale Verhalten der Freiersetzung.

Hier weist die resolutionsbasierte Addition nach, dass durch die Wahl des Ersetzungsschemas eine Art Topologie auf dem Suchraum konstruiert werden kann. Diese Wahl wird hier zum Strategieparameter; sie hat dabei nicht nur Einfluss auf die Dynamik und Problemlösung, sondern auch auf indirekte Attribute des Systems. Nimmt man wie beim logarithmischen Kostenmaß an, dass die benötigte Zeit für eine Addition von der Länge (z. B. in der Binärdarstellung) der Operanden abhängt, so ist hier bei der Eduktersetzung eine deutlich höhere Laufzeit zu erwarten.

8.2 Realisierung einer reaktionsbegleitenden Kostenzählung

Die Lösung eines Optimierungsproblems zeichnet sich im Wesentlichen durch zwei Attribute aus:

1. Diese Lösung muss natürlich das Problem im eigentlichen Sinne lösen und dabei gegebenfalls Randbedingungen beachten. Beim Traveling-Salesman-Problem (TSP) beispielsweise muss eine Reiseroute gefunden werden, die alle Städte berührt und wieder zum Ausgangsort zurückführt.

2. Die Lösung muss bezüglich eines gegebenen Kriteriums optimal sein. Es gibt daher keine andere Lösung aus (1), die dieses Kriterium besser erfüllt. Beim TSP ist das Kriterium die Summe der angefallenen Reisekosten. Optimierung bedeutet dann die Minimierung dieser Kostenfunktion.

Will man ein Optimierungsproblem in einer Künstlichen Chemie formulieren, so sollte jeder Rechenschritt daraufhin angelegt sein, für beide angegebenen Eigenschaften Arbeit zu leisten. Naheliegend ist hier ein zweiteiliger Klauselaufbau. Im ersten Teil der Klausel wird die Lösung des Kernproblems vorangetrieben, im anderen dagegen wird die Güte der aktuellen Teillösung bezüglich des Optimierungskriteriums protokolliert (siehe Abb. 8.6). Für beide Problemsektionen soll dasselbe Reaktionsschema genutzt werden: die Resolution. Bevor dazu im Folgenden das Resolutionsprinzip in natürlicher Weise erweitert wird, noch eine Bemerkung zu der Konstruktion solchermaßen sektionierter Klauseln, *Sektionsklauseln* genannt.

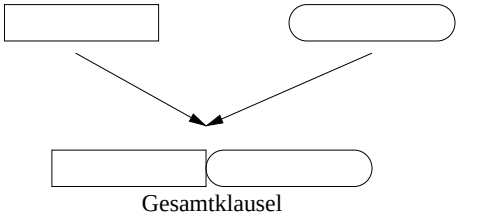


Abbildung 8.6: Aufbau einer Sektionsklausel für Optimierungsprobleme.

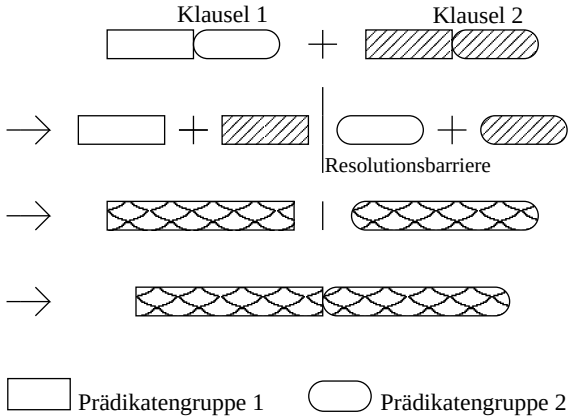


Abbildung 8.7: Separate Multiresolution zwischen Bi-Sektionsklauseln.

Die Aufteilung einer Klausel in zwei (*Bi-Sektionsklauseln*) oder mehr Bereiche kann leicht durch die Wahl disjunkter Prädikate erreicht werden. Dabei ist die Klausel nicht wirklich geteilt, weiterhin sind alle Literale in gleicher Weise ODER-verknüpft. Die virtuelle Aufspaltung dagegen ergibt sich durch die nach Konstruktion (Def. 1.2.12) gegebene Restriktion der Resolutionsanwendung: nur Literale gleichen Prädikatnamens können resolviert werden.

In der herkömmlichen Anwendung des Resolutionsprinzips im automatischen Beweisen (siehe Kapitel 1 sowie Kapitel 6) wird pro versuchter Resolutionsanwendung höchstens 1 Paar von Literalen resolviert. Hier wird aber nun eine Resolution pro Problemsektion erforderlich. Mit anderen Worten werden zwei resolutionsbasierte Rechenschritte auf unabhängigen Klauselbereichen parallel ausgeführt und die Teilschritte im Gesamtresultat wieder kombiniert. Abbildung 8.7 zeigt diesen Prozess, der auf der *separaten Multiresolution* basiert:

Definition 8.2.1 (Separate Multiresolution)

Die *separate Multiresolution* $sMR_{\mathcal{P}_1,\ldots,\mathcal{P}_n}(K_1,K_2)$ auf Prädikatenmengen $\mathcal{P}_1,\ldots,\mathcal{P}_n$ zwischen zwei Sektionsklauseln K_1 und K_2 testet pro Prädikatenmenge $\mathcal{P}_i$ eine binäre Resolution zwischen K_1 und K_2. Nur wenn alle n Resolutionen durchführbar sind, werden diese n Resolutionen auch durchgeführt, und zwar parallel in einem Schritt. Sie münden in eine sektionierte Ergebnisklausel. Die Prädikatenmengen werden *Resolutionssektionen* genannt.

Eine Variante ermöglicht die Priorisierung einer Prädikatenmenge. Diese kann dann alleine eine Resolution in allen Sektionen auslösen, die dies zulassen, *wenn* eine Resolution eine Auflösung der ausgezeichneten Menge zur Folge hat. Durch Unterstreichung der betreffenden Prädikatenmenge wird diese formal gekennzeichnet.

Bemerkung: Die Erweiterung des Resolutionsmechanismus ist deswegen unkritisch, da hier nicht mehr die Semantik der Prädikatenlogik verwendet und *RESAC* nicht als Theorembeweiser eingesetzt wird (siehe auch Unterkapitel 7.1).

Wie die vorgestellten Konzepte der Sektionsklauseln und der separaten Multiresolution kombiniert zur Lösung von Optimierungsproblemen nutzbar gemacht werden können, wird an Hand des Beispiels des TSP im nächsten Kapitel konkret gezeigt.

8.3 Eine resolutionsbasierte Formulierung des Traveling-Salesman-Problems

Dieses Unterkapitel verdeutlicht die Implementierung von Optimierungsproblemen in *RESAC* am Beispiel des Traveling-Salesman-Problems, das als Musterproblem im Optimierungsbereich angesehen wird. Insbesondere wird die Klauselformulierung und die Durchführung der Experimente dargelegt. Abschließend werden Verbesserungen – unter Anderem eine Topologieänderung – vorgeschlagen, die aufgrund ihrer allgemeinen Natur auf andere Probleme übertragbar sind.

Das Traveling-Salesman-Problem (TSP) ist wie folgt definiert.

Definition 8.3.1 (Traveling-Salesman-Problem)
Gegeben sei eine $n \times n$-Matrix $C = (c_{ij})$, die als Distanz- oder Kostenmatrix bezeichnet wird. Die Aufgabe besteht darin, eine Permutation π der natürlichen Zahlen 1 bis n zu finden, die die Summe $\sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)}$ minimiert. Ein anschauliches Bild ist das folgende: Ein Vertreter muss die Städte 1 bis n in einer beliebigen Reihenfolge besuchen. Er versucht dabei, die Länge seiner Reiseroute minimal zu halten.

Das TSP ist eins der fundamentalen Probleme in der kombinatorischen Optimierung; es ist bekannt, dass es NP-hart ist. In der weiteren Betrachtung soll gelten, dass die Kostenwerte natürliche Zahlen sind, angereichert durch das Symbol ∞, dessen Verwendung die Nichtexistenz eines Weges zwischen Städten i und j anzeigt. Diese Erweiterung erhöht die Komplexität des Problems, denn die Künstliche Chemie läuft nun Gefahr, Wege zu wählen, die sich nicht mehr zu einer gültigen Rundreise komplettieren lassen, sich sozusagen in „Sackgassen“ zu verlaufen. Dies wird im Folgenden unter dem Begriff der *Sackgassenbildung* verstanden. Weist ein TSP nur Kostenwerte kleiner ∞ aus, so wird es *barrierefrei* genannt.

8.3.1 Problemrepräsentation

Die Problemstellung beim TSP ist ähnlich der im Unterkapitel 7.2 beschriebenen Suche nach gerichteten Hamilton-Pfaden (DHPP). Sie unterscheidet sich

1. durch die Suche nach geschlossenen Pfaden, so genannten Kreisen – der Vertreter muss zum Ausgangspunkt seiner Reise zurückkehren – und

2. durch die Assoziation eines Kostenwerts zu jeder Kante.

Metapher	graphenth. Entsprechung	*RESAC*-Entsprechung
Stadt	Knoten	Konstantenbezeichner
elementares Wegstück	Kante	Kantenklausel
Ansammlung von $(n-i)$ elementaren Wegstücken	Menge von Kanten	Schablonenklausel$^{(i)}$, syn. Schablone$^{(i)}$
zusammenhängender Weg aus $(n-i)$ elementaren Wegstücken	Pfad über $n-i$ Kanten	Kettenschablone$^{(i)}$
Rundreise	Kreis	Kettenschablone$^{(0)}$

Tabelle 8.1: Begriffe unterschiedlicher TSP-Repräsentationen. n bezeichnet die Anzahl der Städte.

Sprachregelung

Um Konfusionen zu vermeiden, vereinbart Tabelle 8.1 die im Folgenden zugrunde liegende Sprachregelung. Zentral ist der Begriff der *Schablone*. Die Schablone steht für eine Klausel, in der unter Anderem[3] eine Rundreise durch alle n Städte geplant, d. h. kodiert ist. Die folgende Schablone ist ein Beispiel:

$$(X_0 \to X_1),\ (X_1 \to X_2),\ \ldots,\ (X_{n-2} \to X_{n-1}),\ (X_{n-1} \to X_0). \tag{8.7}$$

Die X_i stellen Platzhalter für die einzelnen Städte dar. (8.7) legt keine Stadt fest, wohl aber die Konnektivität (hier: eine Rundreise). Gesucht werden *n elementare Wegstücke*, die jeweils direkte Verbindungen zwischen zwei Städten beschreiben. Trifft[4] diese Schablone auf eine Städteverbindung, also auf ein elementares Wegstück, so wird dieses, falls möglich, in der vorgeplanten Rundreise berücksichtigt. Die Kombination der Schablone aus (8.7) mit der Städteverbindung $(a \to b)$ könnte zu folgender modifizierten Schablone führen:

$$(X_0 \to X_1),\ (X_1 \to a),\ \cancel{(a \to b)},\ (b \to X_4),\ (X_4, X_5),\ \ldots,\ (X_{n-1} \to X_0). \tag{8.8}$$

Die Verbindung $(a \to b)$ hätte in diesem Beispiel auch genauso gut an jeder anderen Position der Rundreise eingesetzt werden können. In jedem Fall aber wird ein elementares Wegstück der Rundreise konkret bestimmt (*instanziiert*), hier das Wegstück $(a \to b)$. Die Streichung in (8.8) macht deutlich, dass es in der weiteren Planung nicht mehr berücksichtigt wird – gesucht werden nur noch $n-1$ elementare Wegstücke. Durch die Instanziierung ergeben sich Anforderungen an die angrenzenden Wegstücke $(X_1 \to a)$ und $(b \to X_4)$. Diese sind jetzt nicht mehr variabel. Dennoch sind sie nicht instanziiert, denn ein Wegstück mit Endpunkt a oder Anfangspunkt b wurde ja (noch) nicht in die Rundreiseplanung eingesetzt. Ein weiteres Beispiel verdeutlicht die Instanziierung: Gegeben ist die Schablone

$$(a \to X_1),\ (X_1 \to X_2),\ (X_2 \to b),\ \ldots \tag{8.9}$$

Erfolgt eine Instanziierung mit Wegstück $(c \to d)$, so ergibt sich die Schablone

$$(a \to c),\ \cancel{(c \to d)},\ (d \to b),\ \ldots \tag{8.10}$$

[3] Eine Schablone kann neben der Kodierung einer Rundreise weitere Informationen enthalten.

[4] Die Terminologie ist vor dem Hintergrund von in einem Reaktor aufeinander treffenden Molekülen zu verstehen. Schablonen und Städteverbindungen sind solche Moleküle, wie später dargelegt wird.

Das Wegstück $(c \rightarrow d)$ ist bestimmt und fällt aus der Planung weg. Die elementaren Wegstücke $(a \rightarrow c)$ und $(d \rightarrow b)$ sind dagegen noch nicht bestimmt (obwohl sie nicht mehr variabel sind) und werden gesucht. Erst eine Instanziierung mit Stadtverbindung $(a \rightarrow c)$ bzw. $(d \rightarrow b)$ entfernt diese aus der Planung.

In einer Schablone sind, wie gesehen, eine bestimmte Anzahl elementarer Wegstücke instanziiert[5], die restlichen werden gesucht. Wieviele elementare Wegstücke einer Rundreise noch nicht bestimmt sind, bezeichnet der durch Indizierung notierte *Freiheitsgrad*. So bezeichnet der Ausdruck *Schablonenklausel*$^{(3)}$ (oder synonym: *Schablone*$^{(3)}$) eine Klausel, die eine Rundreise durch n Städte plant, wobei drei elementare Wegstücke noch nicht instanziiert sind.

Logik-Repräsentation

Um das TSP in *RESAC* zu repräsentieren, kommen zwei Sorten von Klauseln zum Einsatz: die oben angesprochenen Schablonen und die Kantenklauseln, die die Information der Distanzmatrix C übertragen. Befüllt wird der Reaktor ausschließlich mit Klauseln der Menge *Startklauseln* (siehe Kapitel 3), die für ein TSP mit n Städten, bezeichnet durch $i_0, \ldots, i_{n-1}$, konkret die folgenden Klauseln enthält:

- Schablonenklausel$^{(n)}$ – auch *initiale* Schablone genannt –

  $$\overline{\text{Path}(\text{edge}(i_0, X_1))}, \overline{\text{Path}(\text{edge}(X_1, X_2))}, \ldots, \overline{\text{Path}(\text{edge}(X_{n-2}, X_{n-1}))}, \overline{\text{Path}(\text{edge}(X_{n-1}, i_0))}.$$

  O. B. d. A. wird hier Stadt i_0 zum Start- und Zielpunkt der Rundreise gemacht.

- Für jedes elementare Wegstück $(i_k \rightarrow i_l)$ mit assoziierten Kosten c wird die Kantenklausel

  $$\text{Path}(\text{edge}(i_k, i_l)),\ \overline{\text{Visited}(i_k)}, \overline{\text{Visited}(i_l)},\ \overline{\text{Costs}(X, Y_0)}, \ldots, \overline{\text{Costs}(Y_{c-1}, Y_c)}, \text{Costs}(X, Y_c)$$

  aufgenommen.

Im Vergleich zur Klauselrepräsentation des DHPP fällt auf, dass kein Terminierungskriterium vorhanden ist. Es wird durch ein Filterkriterium mit gleichem Aufbau ersetzt:

- Filterkriterium: $\overline{\text{Visited}(i_0)}, \ldots, \overline{\text{Visited}(i_{n-1})}$.

Alle Klauseln werden als Bi-Sektionsklauseln angesehen. Während der eine Bereich der Lösungsfindung dient, werden im anderen Bereich die anfallenden Kosten des aktuellen Lösungskandidaten nachgehalten. Abbildung 8.8 zeigt die Idee. Die zugehörige separate Multiresolution $sMR_{\{\text{Path, Visited}\},\{\text{Costs}\}}$ führt, falls eine Kantenklausel einen Beitrag zur Lösungsfindung liefert, einerseits die Resolution der entsprechenden Path-Literale, andererseits die Resolution der Costs-Literale durch. Nach vorstehender Logikrepräsentation sind nur Resolutionen zwischen Kanten- und Schablonenklauseln möglich.

[5] und werden nicht mehr aufgeführt

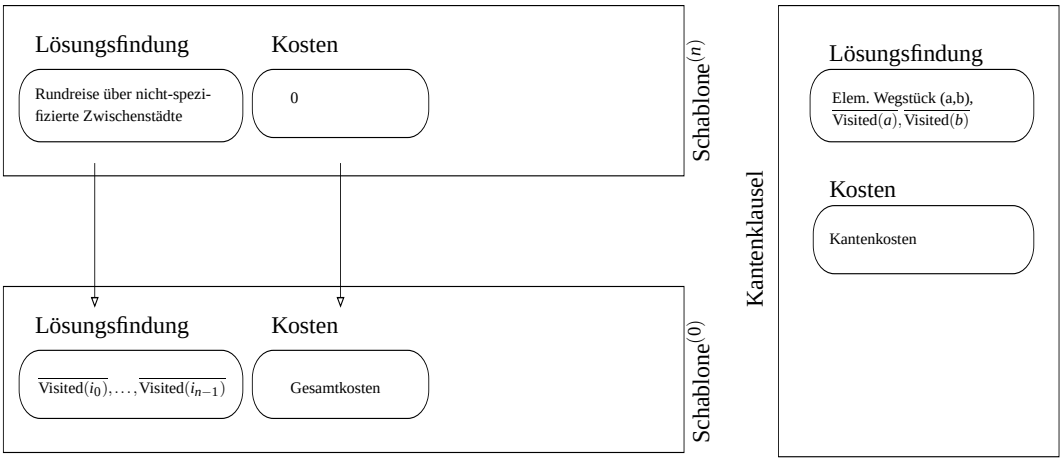


Abbildung 8.8: Bi-Sektionsklauseln für das TSP. In der ersten Sektion wird die Lösungsfindung vorangetrieben. Eine unspezifizierte Rundreise wird durch geeignete Kantenkollisionen schrittweise durch elementare Wegstücke instanziiert. Dabei reichern sich Visited-Literale der besuchten Städte an. Die Kosten der schon instanziierten Wegstücke werden in der Kostensektion nachgehalten.

Beispiel: Es sei $n = 5$ und die Städte durchnummeriert von 0 bis 4. Man betrachte die Reaktion zwischen folgender Kanten- und Schablonenklausel:

$$\begin{aligned}\text{Kantenklausel}\quad & \text{Path}(\text{edge}(4,1)),\ \overline{\text{Visited}(4)}, \overline{\text{Visited}(1)},\\ & \overline{\text{Costs}(X,Y_0)}, \overline{\text{Costs}(Y_0,Y_1)}, \text{Costs}(X,Y_1)\end{aligned}$$

$$\begin{aligned}\text{Schablonenklausel}^{(4)}\quad & \overline{\text{Path}(\text{edge}(0,3))}, \overline{\text{Path}(\text{edge}(2,X_3))}, \overline{\text{Path}(\text{edge}(X_3,X_4))}, \overline{\text{Path}(\text{edge}(X_4,0))},\\ & \overline{\text{Visited}(3)}, \overline{\text{Visited}(2)}, \overline{\text{Costs}(X,Y_0)}, \overline{\text{Costs}(Y_0,Y_1)}, \text{Costs}(X,Y_1)\end{aligned}$$

Resolviert werden die Literale $\text{Path}(\text{edge}(4,1))$ und $\overline{\text{Path}(\text{edge}(X_3,X_4))}$ unter der Substitution $\{X_3/4, X_4/1\}$. Das Ergebnis der separaten Multiresolution ist dann die

$$\begin{aligned}\text{Schablonenklausel}^{(3)}\quad & \overline{\text{Path}(\text{edge}(0,3))}, \overline{\text{Path}(\text{edge}(2,4))}, \overline{\text{Path}(\text{edge}(1,0))}, \overline{\text{Visited}(3)}, \overline{\text{Visited}(2)},\\ & \overline{\text{Visited}(4)}, \overline{\text{Visited}(1)}, \overline{\text{Costs}(X,Y_0)}, \overline{\text{Costs}(Y_0,Y_1)}, \overline{\text{Costs}(Y_1,Y_2)}, \text{Costs}(X,Y_2)\end{aligned}$$

Die elementare Wegstrecke $(4,1)$ hätte in diesem Fall keine andere elementare Wegstrecke ersetzen können; es ist keine andere Resolution möglich. Im Allgemeinen wird es jedoch mehrere Möglichkeiten geben.

8.3.2 Durchführung eines Experiments

Füllung und Betrieb des gut gerührten Zuflussreaktors geschieht auf gewohnte Weise (siehe Algorithmus 3.3). Allerdings wird die separate Multiresolution verwendet. Kanten- und Schablonenklauseln$^{(n)}$ fließen in den Reaktor. Treffen im Reaktorbetrieb passende Kanten und

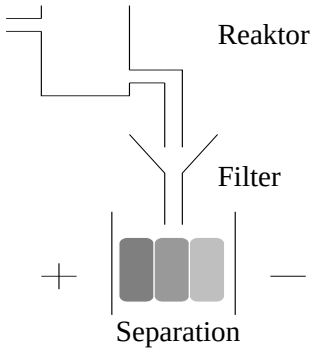


Abbildung 8.9: Durchführung einer Optimierung. Die Nachverarbeitung teilt sich auf in Filterung und Auftrennung der Klauseln. Die hier gezeigte Separation ähnelt einer Gelelektrophorese.

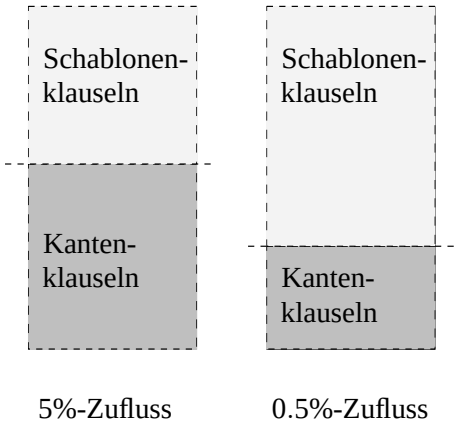


Abbildung 8.10: Konzentrationsverteilung im Reaktor. Die Abbildung zeigt das Verhältnis von Kanten- zu Schablonenklauseln zu einem TSP-Experiment mit Eduktersetzung bei unterschiedlichem Zufluss.

Rundreiseplanungen aufeinander, entstehen wiederum Schablonenklauseln, allerdings mit erniedrigtem Freiheitsgrad. Es gibt im Experiment kein Terminierungskriterium wie bei den DHPP-Experimenten. Stattdessen wird die Rührung nach einer vorgegebenen Zeit unterbrochen und eine Nachverarbeitungsphase eingeleitet.[6]

Die Nachverarbeitung besteht aus zwei Prozessschritten:

(Filterung) Abgleich mit dem Filterkriterium. Nur die Klauseln, die dem Kriterium entsprechen, passieren das Filter.

(Separation) Auftrennung der Klauseln nach Gewicht oder Anzahl der Costs-Literale.

Nach dem ersten Schritt der Nachverarbeitung gibt es nur noch Schablonenklauseln, die tatsächlich Lösungen repräsentieren. Unter diesen Lösungen wird im zweiten Schritt eine Lösung mit minimalen Kosten bestimmt. Eine mögliche Versuchsanordnung außerhalb von Simulationen zeigt Abbildung 8.9. Vorbild für die Durchführung der zweiten Phase ist die Technik der Gelelektrophorese, bei der in einem elektrischen Feld einzelne DNA-Stränge nach ihrer Größe sortiert und anschließend der Länge nach aufgetrennt werden. Herkömmliche Sedimentierung ist eine andere Möglichkeit, die Separation durchzuführen.

8.3.3 Grobanalyse des Systems

Der vorstehende Systementwurf weist zwei schwerwiegende Nachteile auf:

[6] In der Simulation kann diese Verarbeitung auch durch kontinuierliche Beobachtung (monitoring) simultan erfolgen. Ausgefilterte Klauseln werden in diesem Fall durch Zufluss einer neuen Klausel ersetzt.

Ungünstige Konzentrationsverteilung

Die entscheidende Rolle im Experimentaufbau hat der Reaktor. Der Reaktorinhalt verteilt sich auf Kantenklauseln und Schablonenklauseln. Letztere liegen in unterschiedlichen „Einsetzungszuständen“ vor: von der initialen Schablone$^{(n)}$, die Rundreisen der Form nach vorzeichnet und noch kein Wegstück bestimmt hat, bis hin zur Schablone$^{(0)}$, in der alle Wegstücke instanziiert sind.

Erste Experimente zeigen (siehe Abb. 8.10), dass Kantenklauseln einen großen Teil des Reaktors ausmachen. Produktive Kollisionen treten jedoch nur dann auf, wenn Kantenklauseln mit Schablonen zusammentreffen. Die Bildung der Rundreisen verläuft so (in der Simulation) sehr ineffizient. In dem in Abb. 8.10 links gezeigten Beispiel sind mindestens $(0,54)^2 + (0,46)^2$, also mehr als 50% der Kollisionen von vorneherein, d. h. konstruktionsbedingt, elastisch. Dieser Grundanteil unproduktiver Kollisionen ist problemunabhängig.

Die Eigenschaft scheint systemimmanent zu sein und als solche nicht leicht zu verändern, da erstens Zufluss an Kantenklauseln absolut notwendig ist, und zweitens eine Verschiebung des Konzentrationsverhältnisses von Kantenklauseln zu Schablonenklauseln keine spürbare Verbesserung bewirkt. Ein Verhältnis von 50 : 50 ist optimal im Sinne der Minimierung des Grundanteils elastischer Kollisionen. Dieses optimale Verhältnis bedeutet aber eine große Platzverschwendung, tatsächlich steht so nur die Hälfte des Reaktors den Schablonen – also den Lösungskandidaten – zur Verfügung.

Ungültige Rundreisen

Eine Aufschlüsselung der im Reaktor entstehenden Schablonen zeigt die Entstehung vieler ungültiger Rundreisen. Dieser Umstand wiegt besonders schwer in der zeit- und platzaufwändigen Simulation. Verfolgt man eine Schablone startend im Zustand Schablone$^{(n)}$ bei ihrer schrittweisen Modifizierung, so lassen sich zwei Fälle unterscheiden: (1) Schablone$^{(0)}$ wird nie erreicht oder (2) Schablone$^{(0)}$ wird erreicht.

Fall 1: Im ersten Fall kann keine elementare Wegstrecke mehr in die Schablone eingesetzt werden, obwohl noch nicht alle Wegstücke instanziiert sind. Einerseits ist es möglich, dass eine „Sackgasse“ erreicht wurde: Durch vorangegangene Festlegungen ist eine Situation eingetreten, in der ganz bestimmte Wegstücke nachgefragt werden, die die Problemstellung jedoch nicht bereitstellt – nicht zwischen allen Städten gibt es in nicht-barrierefreien Problemen einen Weg. Dieser Aspekt wird später noch einmal aufgegriffen (Unterkapitel auf Seite 156). Andererseits können sich Zwangspunkte durch Einsetzung von Wegstücken an ungünstigen Positionen ergeben. Dies untersucht der nächste Abschnitt.

Eine ungünstige Instanziierung elementarer Wegstücke demonstriert die folgende Planung einer Rundreise durch fünf Städte, bezeichnet durch a, b, c, d und e. Abbildung 8.11 zeigt den Verbindungsgraphen. In verkürzter Notation[7] könnte die initiale Schablone – Schablone$^{(5)}$ – so aussehen:

$$(a, X_2)\ (X_2, X_3)\ (X_3, X_4)\ (X_4, X_5)\ (X_5, a).$$

[7] Jeder Klammerausdruck steht für eine elementare Wegstrecke. Abstrahiert wird von den Visited- und Costs-Literalen, außerdem von der Polarität der Path-Literale.

Trifft diese Schablone auf Kanten $e_1 = (b,c)$ und $e_2 = (d,e)$, so könnten diese elementaren Wegstrecken an die 2. bzw. 4. Position der Rundreise gesetzt werden mit folgendem Ergebnis:

$$\underline{(a,b)}\ \underline{\sout{(b,c)}}\ \underline{(c,d)}\ \underline{\sout{(d,e)}}\ \underline{(e,a)}.$$

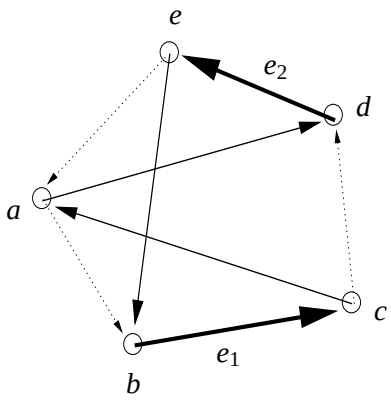


Abbildung 8.11: Problemgraph eines TSP mit 5 Städten. Die durchgezogenen Kanten markieren die gegebenen Wege. Nach Wahl der elementaren Wege e_1 und e_2 werden möglicherweise die strichlierten Kanten gesucht.

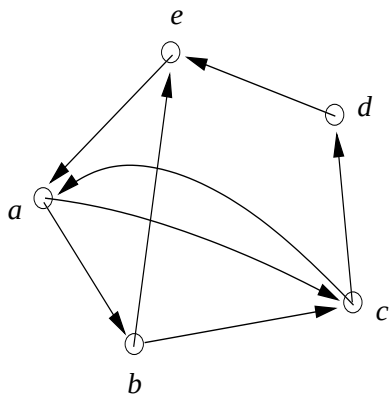


Abbildung 8.12: Das durch diesen Graph definierte TSP gibt die Möglichkeit zur Konstruktion ungültiger Rundreisen. Erklärung siehe Text.

Die durchgestrichenen Wegstrecken fallen aus der neuen Rundreiseplanung, die unterstrichenen werden nun zur Instanziierung gesucht. Da diese aber in der Problembeschreibung nicht spezifiziert sind, wird diese Klausel den Status Schablone$^{(2)}$ nie erreichen.

Fall 2: Schablone$^{(0)}$ wird erreicht. In diesem Fall konnten alle elementaren Wegstrecken der Schablone instanziiert werden. Nicht alle erzeugten Rundreisen sind jedoch auch gültig, wie mit Abbildung 8.12 gezeigt werden kann. Aus der dort visualisierten Problemstellung entsteht möglicherweise die Rundreise:

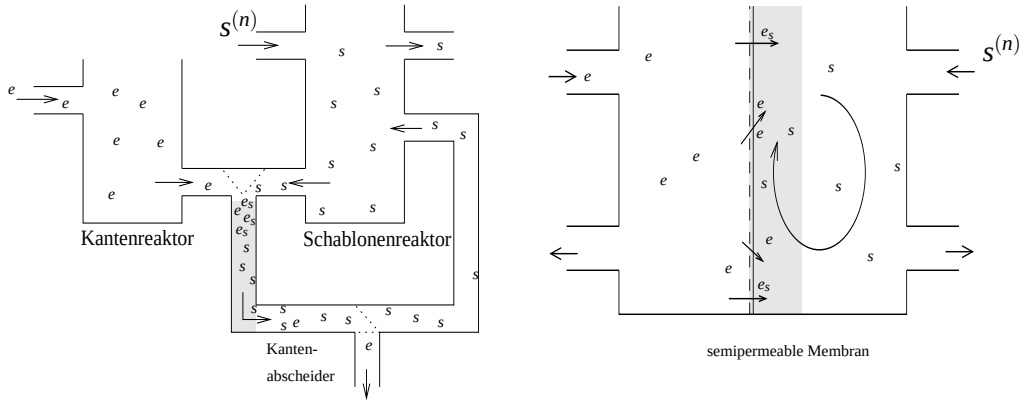
$$a \longrightarrow b \longrightarrow e \longrightarrow a \longrightarrow c \longrightarrow a.$$

Grund für unzulässige Rundreisen ist die Mehrfacheinbindung von Städten. Zwar werden solche „Lösungen“ ausgefiltert, doch wäre es wünschenswert, solche Schablonen erst gar nicht entstehen zu lassen, da diese Platz verbrauchen ohne zur Optimierung beizutragen.

8.3.4 Verbesserung des Entwurfs

Änderung der Reaktortopologie

Um den ersten Nachteil des bisherigen Modells, nämlich die ungünstige Konzentrationsverteilung, zu beseitigen, ist eine Veränderung des Reaktoraufbaus angezeigt. Dieses Unterkapitel stellt



Doppelreaktorsystem Zweibereichsreaktor

Abbildung 8.13: Beispiele alternativer Reaktoranordnungen. Gemeinsam ist die Trennung von Kanten- und Schablonenklauseln (bezeichnet durch e und s). Reaktionen (hier angedeutet durch e_s) finden nur im Reaktionsterrain statt (im Bild schraffiert). Zugeführt werden Kantenklauseln und Schablonen$^{(n)}$. Weitere Erklärung siehe Text.

ein neues Modell vor, das beispielsweise als Doppelreaktorsystem oder Zweibereichsreaktor ausgeführt werden kann. Abbildung 8.3.4 zeigt beide Anordnungen. Das Entscheidende ist die Trennung von Kanten- und Schablonenklauseln, zwei separierte Bereiche entstehen: der Schablonen- und der Kantenreaktor.

Kantenklauseln werden aus ihrem angestammten Aufenthaltsbereich ausgeführt in ein *Reaktionsterrain*, das außerhalb beider Reaktorbereiche oder auf dem Gebiet der Schablonenklauseln liegt. Dort werden sie schnell mit Schablonenklauseln „abreagiert“. Die entstehenden Schablonen werden dem Schablonen-Reservoir wieder zugeführt beziehungsweise verbleiben dort. Kanten- und Schablonenklauseln werden durch geregelten Zufluss den jeweiligen Reaktorbereichen kontinuierlich zugeführt. Für eine Simulation genügt es nun, nur noch Kollisionen im Reaktionsterrain zu betrachten.

Die sich aus dieser Topologie ergebenden Vorteile sind wie folgt:

(1) Bezüglich ihres Typs zueinander passende Kollisionspartner werden gezielt zusammengeführt mit der Folge, dass sich Schablonen schnell mit Wegstücken sättigen.

(2) Der die Kantenklauseln beherbergende Reaktorteil kann relativ klein gehalten werden; es ergibt sich eine erhebliche Platzeinsparung. Diese kann zu Gunsten einer großen Schablonenpopulation genutzt werden, was die Vorhaltung vieler Lösungskandidaten ermöglicht.

Schablonen-Endersetzung

Der in der Analyse des ursprünglichen Entwurfs ausgemachte zweite Nachteil – die Bildung vieler ungültiger Rundreisen – kann durch einen alternativen Lösungsprozess abgemildert werden. Dieser beinhaltet, dass Kantenklauseln nur mit den Enden der Schablonen reagieren können. Dadurch entstehen zusammenhängende Wegstücke, da Anfangs- und Endpunkt einer Rundreise

schon durch die initiale Schablone$^{(n)}$ o. B. d. A. bestimmt sind und sich einander entsprechen. Der Resolutionsprozess sichert im Fortgang des Experiments, dass jede Schablone einen festgelegten Anfangs- oder Endpunkt hat. Kanten mit passendem Randpunkt vervollständigen bei Kollision diese Schablonen.

Elementare Wegstücke werden so nicht mehr an unsinnigen Stellen der Rundreise eingesetzt. Jede Schablone steht für eine Rundreise, in der schon ein mehr oder weniger langes, aber immer *zusammenhängendes* Wegstück festgelegt wurde. Solche Schablonen werden *Kettenschablonen* genannt.

Beispiel: (Kettenschablonen)

Zu dem in Abb. 8.11 gegebenen Graphen wird sich eine Kettenschablone vom Typ Schablone$^{(5)}$ mit Anfangsstadt *a* unter den nachstehend aufgeführten Kollisionen wie folgt entwickeln (die eine Resolution ermöglichenden Literale sind unterstrichen, außerdem wird wieder die vereinfachte Notation angewendet):

$\underline{(a,X_2)}\ (X_2,X_3)\ (X_3,X_4)\ (X_4,X_5)\ \underline{(X_5,a)}$	Schablone$^{(5)}$	Kollision mit (a,d)
$\sout{(a,d)}\ \underline{(d,X_3)}\ (X_3,X_4)\ (X_4,X_5)\ \underline{(X_5,a)}$	Schablone$^{(4)}$	Kollision mit (c,a)
$\underline{(d,X_3)}\ (X_3,X_4)\ \underline{(X_4,c)}\ \sout{(c,a)}$	Schablone$^{(3)}$	Kollision mit (b,c)
$\underline{(d,X_3)}\ \underline{(X_3,b)}\ \sout{(b,c)}$	Schablone$^{(2)}$	Kollision mit (d,e)
$\sout{(d,e)}\underline{(e,b)}$	Schablone$^{(1)}$	Kollision mit (e,b)
$\sout{(e,b)}$	Schablone$^{(0)}$	

Überlegung zur Realisierung der Endersetzung in einem Modell des molekularen Rechnens

Eine der Möglichkeiten, das Verfahren der Endersetzung in ein reales System abseits der Simulation zu übertragen, basiert auf einer speziellen Sekundärstruktur der Moleküle (siehe Abb. 8.14). Hierbei sind die Kettenschablonen so verformt, dass sich nur deren reaktive Enden zur Reaktion anbieten.

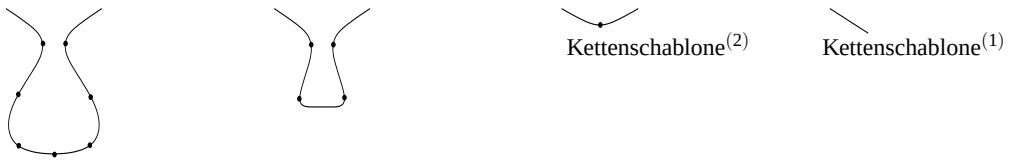


Abbildung 8.14: Angebot der Endliterale zur Resolution durch eine geeignete Sekundärstruktur. Im Bild sind Literale durch Punkte getrennt.

Eine andere Möglichkeit basiert auf dem Umstand, dass nur die reaktiven Literale am Rand Konstantenbezeichner – also schon festgelegte Städte – aufweisen. Enzyme, die auf Marker reagieren, die allen Konstanten angefügt sind, können den Resolutionsprozess auf die randständigen Literale lenken.

Vermeidung von Mehrfacheinbindungen von Zwischenstädten

Wird verhindert, Zwischenstädte mehrfach einzubinden, reduziert sich die Anzahl unzulässiger Lösungen: Darf nur der Ort, der Ausgangs- und Zielort der Rundreise markiert, zweimal besucht werden, die restlichen Städte dagegen nur genau einmal, so sind unzulässige Rundreisen ausgeschlossen. Sofern eine Kettenschablone$^{(0)}$ hergeleitet wird, repräsentiert diese immer eine gültige Rundreise.

Bemerkung: Natürlich bleibt es weiterhin möglich, dass die Einsetzung eines Wegstücks an sich zur Sackgassenbildung führt, eine geplante Rundreise also nicht instanziiert, d. h. verwirklicht werden kann.

Realisierung der Nicht-Mehrfacheinbindung

In *RESAC* stehen nur resolutionsbasierte Reaktionsmechanismen zur Verfügung. Es kann daher kein direkter, intuitiver Test der Form „Wenn Stadt X schon einmal besucht wurde, …“ erfolgen. Stattdessen wird erneut auf das Konzept der separaten Multiresolution zurückgegriffen. Die folgende Konstruktion modifiziert leicht die bisher dargestellte Konstruktion der Kanten- und Schablonenklauseln. Die beiden zu den Endpunkten eines elementaren Wegstücks gehörenden Visited-Literale der Kantenklauseln bleiben bestehen, werden allerdings einer eigenen Sektion mit Namen *Stadtintegration* zugeordnet. Schließlich wird auch die (Ketten-)Schablone$^{(n)}$, die der Menge der Startklauseln angehört, um diese neue Sektion durch Hinzunahne der $(n-1)$ Literale

$$\text{Visited}(i_1), \ldots, \text{Visited}(i_{n-1})$$

erweitert. Für die Konstruktion ist es bedeutsam, den in der Schablone festgelegten Start- und Zielort auszunehmen, hier bezeichnet durch i_0. Abbildung 8.15 zeigt den schematischen Aufbau der Tri-Sektionsklauseln. Aus der unterbundenen Mehrfacheinbindung von Zwischenstädten folgt die unbedingte Gültigkeit einer jeden Kettenschablone$^{(0)}$. Das Filterkriterium (siehe Unterkapitel 8.3.1) ist damit obsolet. Alle Kettenschablonen mit leerer *Lösungsfindungs*-Sektion werden zur Gütebestimmung nach Länge ihrer *Kosten*-Sektion aufgetrennt.

Erlaubt man eine produktive Reaktion nur dann, wenn

entweder in allen Sektionen eine Resolution durchgeführt werden kann (also insbesondere auch in der Sektion *Stadtintegration*)

oder eine Resolution in der Sektion *Lösungsfindung* diese auflöst, also eine komplette und gültige Rundreise hergeleitet wurde,

so wird garantiert, dass Zwischenstädte nicht mehrfach eingebunden werden. Die erstmalige Verwendung einer Stadt streicht das entsprechende Visited-Literal. Der angewendete Resolutionsmechanismus entspricht einer separaten Multiresolution gemäß Def. 8.2.1, formal notiert als $sMR_{\underline{\{\text{Path}\}},\{\text{Costs}\},\{\text{Visited}\}}$. Ein Beispiel im Anhang A demonstriert die Vorgehensweise. Dort wird auch gezeigt, wie die Wegstücke einer Rundreise protokolliert werden können.

Im restlichen Kapitel wird nur noch von Kettenschablonen mit Vermeidung von Mehrfacheinbindungen ausgegangen.

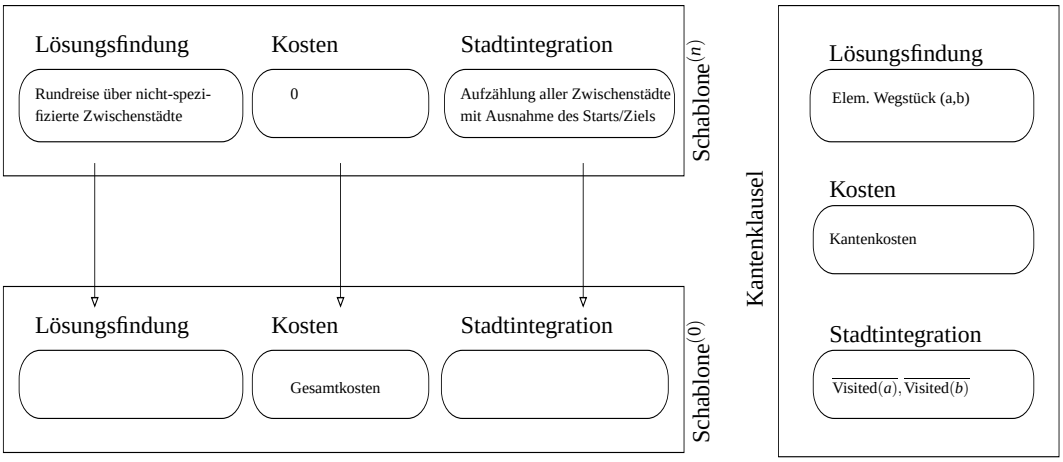


Abbildung 8.15: Tri-Sektionsklauseln für das TSP. In der ersten Sektion wird die Lösungsfindung vorangetrieben. Eine unspezifizierte Rundreise wird bei geeigneter Kantenkollision schrittweise durch elementare Wegstrecken instanziiert, wenn die dabei zu integrierenden Städte nicht beide vorher schon integriert wurden. Die Sektion *Stadtintegration* gibt in jeder Schablone die noch nicht-integrierten Städte an.

Entfernung nicht-reaktiver Schablonen und gesteuerter Zufluss

Wie schon bemerkt, schließen die vorgenannten Verbesserungen nicht aus, dass geplante Rundreisen durch die Instanziierung mit konkreten Wegstücken nicht mehr zu vollständigen Rundreisen ausgebaut werden können (Sackgassenbildung). Diese unfertigen Kettenschablonen zeichnen sich durch ihre Unfähigkeit aus, weitere Reaktionen einzugehen. Es ist von Vorteil, diese nach einer vorgegebenen Zahl von elastischen Kollision, der *Inaktivitätstoleranz* κ, aus dem Reaktor zu entfernen und durch Zufluss mit initialen Schablonen zu ersetzen.[8]

Der Parameter κ wirkt sich in diesem Zusammenhang auch auf das Verhältnis von Kettenschablonen verschiedener Freiheitsgrade (Schablone$^{(n)}$ bis Schablone$^{(0)}$) untereinander aus. Nimmt man die Perspektive einer Schablone$^{(i)}$ für $0 < i \leq n$ ein, so führt nur[9] die Kollision mit einer passenden Kante zu ihrer Modifikation, nämlich der Transformation zu einer Klausel des Typs Kettenschablone$^{(i-1)}$. Vorausgesetzt es finden sich genug passende Kantenklauseln im Reaktor, durchläuft eine Kettenschablone daher sämtliche Freiheitsgrade von n (Startklausel, initiale Schablone) bis 0 (Lösung). Abbildung 8.16 zeigt den Lebenslauf einer Kettenschablone. Höhere Werte von κ sorgen für längere Verweilzeiten im Reaktor mit dem Nachteil, dass auch nutzlose Schablonen lange verweilen. Auf der anderen Seite verringert ein kleiner Wert die Chancen einer Schablone zur Lösung „heranzureifen“. Übertragend gesprochen ist jede Schablonenklausel auf Bewährung im Reaktor. Die Bewährungszeit wird festgelegt durch den Parameter κ. Verminderung des κ-Werts verschiebt den Konzentrationsschwerpunkt im Schablonenreaktor in Richtung von Schablonen mit hohen Freiheitsgraden, eine Erhöhung in Richtung niedriger Grade.

[8] Man könnte sich hier vorstellen, dass jede elastische Reaktion einen Energieverlust bewirkt.

[9] abgesehen von einer möglichen Ersetzung durch Zufluss

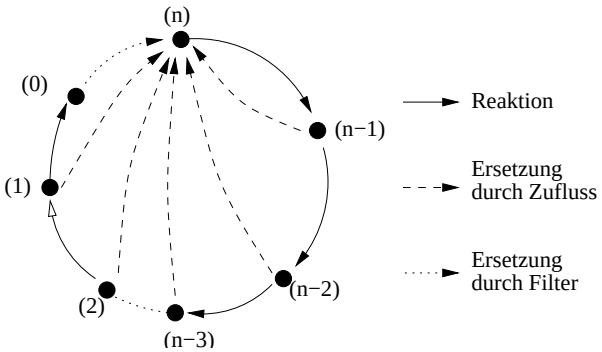


Abbildung 8.16: Lebenszyklus einer Kettenschablone. Als Startklausel mit Freiheitsgrad *n* fließt sie in den Reaktor. Verminderung des Grades durch Reaktion. Ersetzung durch globalen Zufluss oder aufgrund von Inaktivität (lokaler Zufluss). Ersetzung einer Lösung (Grad 0) durch das Filter.

Neben dem durch κ gesteuerten und dem durch ausgefilterte Klauseln (Lösungen) verursachten *außerordentlichen* Zufluss ist für die Konzentrationsverteilung im Schablonenreaktor der reguläre Zufluss entscheidend, gesteuert durch die Zuflussrate *i*. Während die Ersetzung aufgrund von Inaktivität lokal erfolgt – genau die inaktive Schablone wird ersetzt –, so erfolgt der reguläre Zufluss global, d. h. jede Schablonenklausel kann ersetzt werden. Ein hoher Zufluss zerstört so neben inaktiven Schablonen auch hoffnungsvolle Lösungen. Geringer Zufluss löst nur einen niedrigen globalen Druck auf die Schablonenklauseln im Reaktor aus.[10] Erfolgreiche Klauseln können sich so sicher bis zur Kettenschablone$^{(0)}$, einer Lösung, weiterentwickeln. Der Nachteil in nicht-barrierefreien Systemen ist die mögliche Blockade des Systems durch Sackgassenbildung.

In Tabelle 8.2 wird der Zusammenhang der beiden Parameter diskutiert. Abb. 8.18 gibt am Beispiel eines nicht-barrierefreien TSP mit 15 Städten einen Überblick über die aus verschiedenen Parameterkombinationen resultierenden Konzentrationsverläufe. Ein anderes Problem wird unter dem gleichen Aspekt in Abb. 8.19 untersucht. Es handelt sich um ein barrierefreies 53 Städte-Problem aus der *TSPLIB* (Reinelt 1991).

Eine Zuflussstrategie tritt hier erstmals auf, der Threshold-Zufluss. Dabei wird versucht, die Konzentration der initialen Schablone auf ein bestimmtes Level einzupegeln und dieses konstant zu halten. Der Zufluss wird dann gestoppt, wenn die Konzentration dieser Klausel im Reaktor oberhalb des vorgegebenen Schwellenwerts Θ liegt. Fällt sie unter Θ, erfolgt der Zufluss. Die Konzentrationsmessung kann dabei einfach im Abfluss des Reaktors erfolgen (siehe Abb. 8.17); im gut gerührten Reaktor stimmen die Molekülkonzentrationen im Innern des Reaktors mit denen im Abfluss überein (Denbigh und Turner 1973).

[10] Der lokale Druck durch die Inaktivitätstoleranz bleibt davon unberührt.

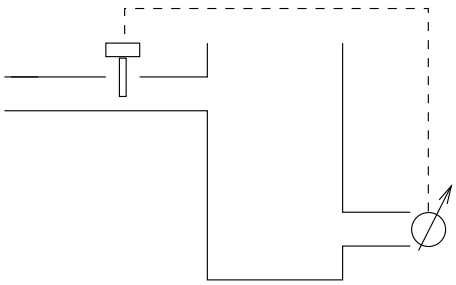


Abbildung 8.17: Threshold-Zufluss. Die Konzentration von Molekülsorten wird durch kontinuierliche Messung und entsprechenden Zufluss gesteuert.

- Erhöhung des Zuflusses „verschiebt“ Freiheitsgrade der Schablonen in Richtung höherer Werte.
- Erhöhung der Inaktivitätstoleranz „verschiebt“ Grade in entgegengesetzter Richtung.

Zuflussstrategie			Bemerkung
kein Zufluss	κ klein	$\ominus$	Schablonen können nicht heranreifen.
	κ mittel	$\oplus$	Inaktive Schablonen werden i. A. ersetzt ($\rightarrow$ 1. Quelle für Zufluss), trotzdem haben die meisten Schablonen ausreichend Zeit, zu Lösungen heranzureifen ($\rightarrow$ 2. Quelle für Zufluss).
	κ hoch	$\ominus$	In nicht-barrierefreien Systemen kann das System durch die Barrieren blockiert werden, der Reaktor ist in diesem Fall absolut geschlossen.
geringer unbedingter Zufluss (z. B. mit Rate 0.5%)		$\bigcirc$	Verbesserung in nicht-barrierefreien Systemen gegenüber absolut geschlossenen Reaktoren, ansonsten Verschlechterung.
mittlerer bis hoher unbed. Zufluss (z. B. mit Rate $\geq 5\%$)		$\ominus$	Unabhängig von κ: globaler Druck zu groß, Schablonen reifen nicht heran.
geringer bedingter Zufluss (z. B. $\theta = 0.5\%$)		$\oplus$	Ausgleichsstrategie: In nicht-barrierefreien Systemen können Sackgassen überwunden werden, ohne dass κ sehr niedrig gewählt werden muss.

Tabelle 8.2: Diskussion der Parameter Zufluss (unbedingt oder bedingt, d. h. Threshold-gesteuert) und Inaktivitätstoleranz κ. Vergleiche dazu Abbildungen 8.18 und 8.19.

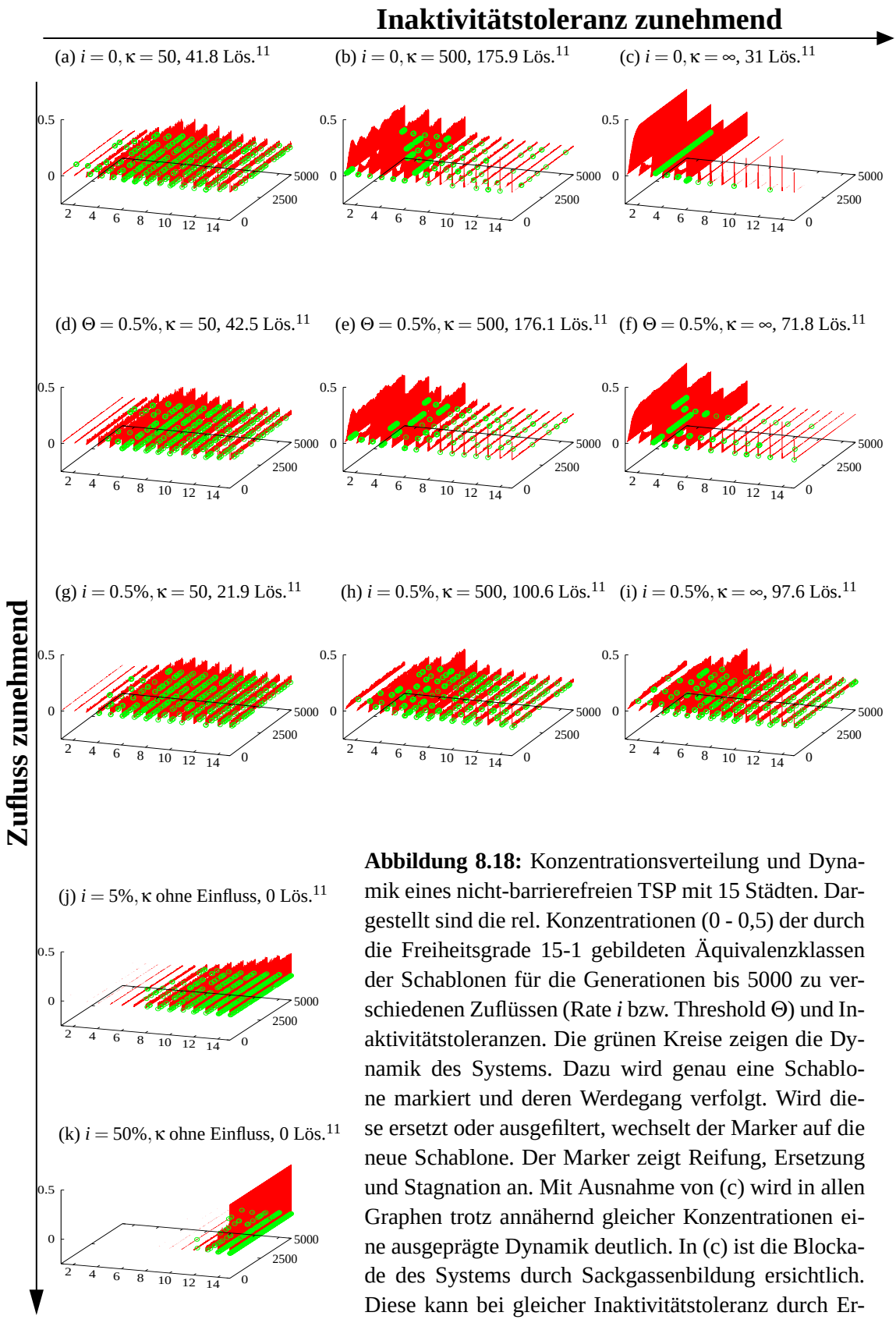


Abbildung 8.18: Konzentrationsverteilung und Dynamik eines nicht-barrierefreien TSP mit 15 Städten. Dargestellt sind die rel. Konzentrationen (0 - 0,5) der durch die Freiheitsgrade 15-1 gebildeten Äquivalenzklassen der Schablonen für die Generationen bis 5000 zu verschiedenen Zuflüssen (Rate i bzw. Threshold Θ) und Inaktivitätstoleranzen. Die grünen Kreise zeigen die Dynamik des Systems. Dazu wird genau eine Schablone markiert und deren Werdegang verfolgt. Wird diese ersetzt oder ausgefiltert, wechselt der Marker auf die neue Schablone. Der Marker zeigt Reifung, Ersetzung und Stagnation an. Mit Ausnahme von (c) wird in allen Graphen trotz annähernd gleicher Konzentrationen eine ausgeprägte Dynamik deutlich. In (c) ist die Blockade des Systems durch Sackgassenbildung ersichtlich. Diese kann bei gleicher Inaktivitätstoleranz durch Erhöhung des Zuflusses überwunden werden, Abb. (f).

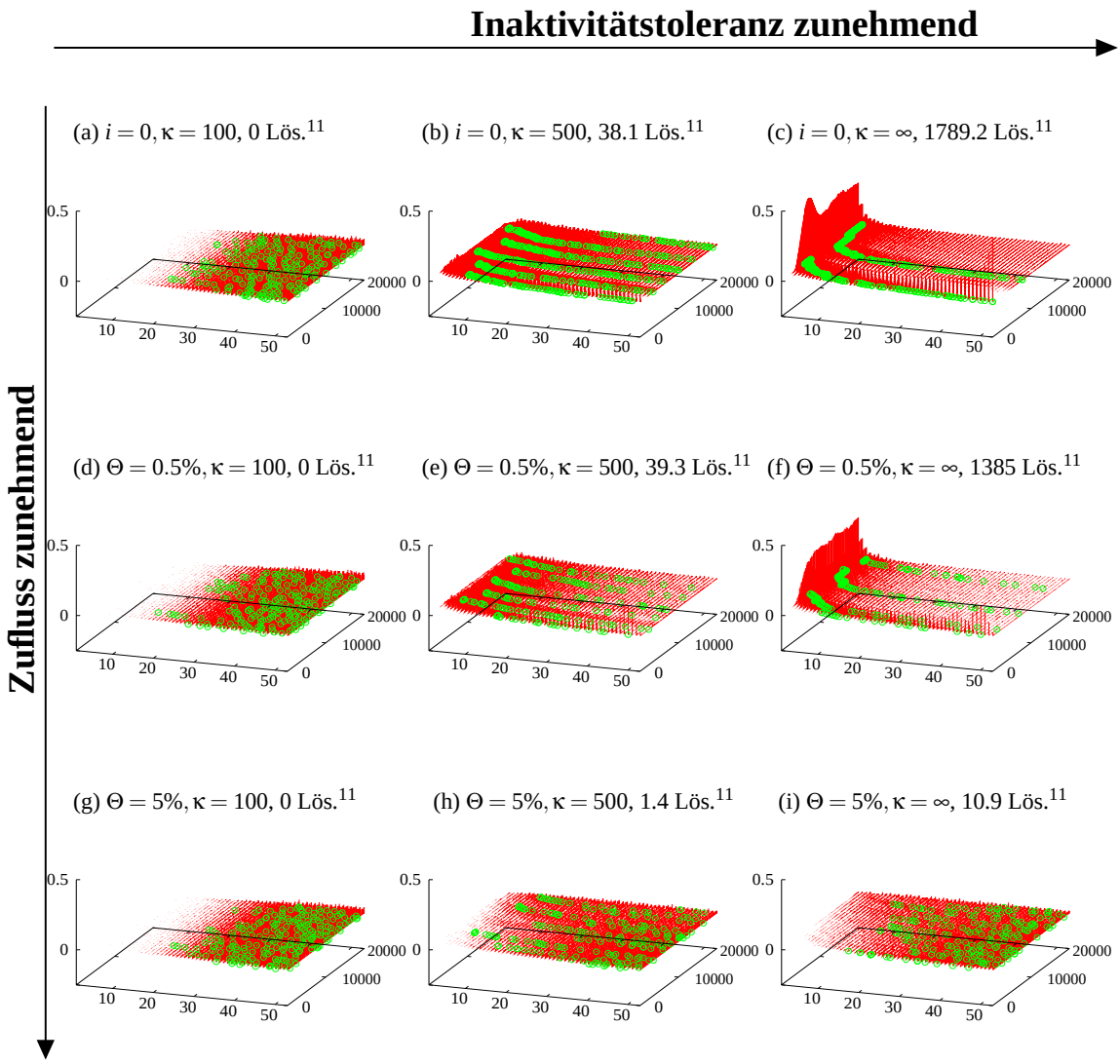


Abbildung 8.19: Konzentrationsverteilung und Dynamik zu einem barrierefreien TSP mit 53 Städten („ft53“ aus der *TSPLIB*). Dargestellt sind die relativen Konzentrationen im Bereich 0-0,5 der durch die Freiheitsgrade 53-1 gebildeten Äquivalenzklassen der Schablonen für die Generationen 1 bis 20000 zu verschiedenen Zuflüssen (Rate i bzw. Threshold Θ) und Inaktivitätstoleranzen. Die grünen Kreise zeigen die Dynamik des Systems. Dazu wird genau eine Schablone markiert und deren Werdegang verfolgt. Wird diese ersetzt oder ausgefiltert, wechselt der Marker auf die neue Schablone. Abb. (c) zeigt hier eine Dynamik, die im nicht-barrierefreien System nicht zu beobachten war. Auch Reaktoren ohne Zufluss mit großer Inaktivitätstoleranz liefern viele Lösungen, da Sackgassenbildung nicht möglich ist. Nach dem Start entwickeln sich alle Schablonen schnell weiter, hohe Freiheitsgrade sind dann zunehmend nicht mehr vorhanden. Erst wenn die ersten Lösungen auftreten (und in der Folge fortwährend) beginnen neue Lebenszyklen durch den durch Ausfilterung hervorgerufenen Zufluss. Der Threshold-Zufluss sorgt für eine relativ gleichmäßige Verteilung der Schablonen (bezüglich ihrer Freiheitsgrade).

[11] Anzahl der gefundenen TSP-Lösungen. Durchschnitt aus 20 Experimenten, auf Zentel gerundet.

8.3.5 Experimente

In *RESAC* wurden die folgenden Traveling-Salesman-Probleme untersucht:

- „007aa“ [12] (7 Städte, asymmetrisch),
- „Sym10“ [13] (10 Städte, symmetrisch),
- „012aa“ [12] (12 Städte, asymmetrisch),
- „nbf15“ (15 Städte, nicht-barrierefrei),
- „ulysses“ [14] (16 Städte, Rundreise des Odysseus (Grötschel und Padberg 1993)) und
- „ft53“ [14] (53 Städte, asymmetrisch)

Die zugehörigen Kostenmatrizen sind im Anhang B aufgelistet, die Ergebnisse der *RESAC*-Experimente in Tabelle 8.3 dargestellt. Während für Probleme mit relativ wenigen Kanten optimale Resultate erzielt wurden, konnten die optimalen Kosten für höher dimensionierte Probleme nur approximiert werden. Die Approximation verschlechtert sich mit zunehmender Kantenzahl. Hier reichte der zur Verfügung stehende Reaktorplatz nicht mehr aus, in angemessener Zeit eine optimale Lösung zu finden. Es konnten nicht genügend viele Zwischenschritte in ausreichender Anzahl gebildet werden.

Problem	Knoten	Kanten	Opt. Kosten	*RESAC*-Lösung (vor Gen.)	Bemerkung
007aa	7	42	1649	1649 (16)	(a)
Sym10	10	90	10003	10003 (102)	(b)
012aa	12	132	1799	1799 (8400)	(b)
nbf15	15	101	9812	9812 (2500)	(b)
ulysses	16	240	6859	7122 (8300)	(b)
ft53	53	2756	6905	21568 (10624)	(b)
(a)	Schablonen-Endersetzung, Ein-Reaktor-System				
(b)	Schablonen-Endersetzung, Zweibereichsreaktor				

Tabelle 8.3: TSP-Experimente in *RESAC*. Die Generation, in der die beste Lösung gefunden wurde, ist von untergeordnetem Interesse, es lassen sich daraus keine Aussagen über andere Experimentverläufe desselben Problems herleiten.

Die Simulationen sind sehr zeit- und/oder speicherplatzaufwändig. Die Stärken dieser resolutionsbasierten künstlichen Chemie zeigen sich, wenn tatsächlich eine sehr große Zahl von Kollisionen in kurzer Zeit stattfinden. DNA-Computing bietet beispielsweise die Möglichkeit, etwa 10 Billionen Operationen in einem Reagenzglas DNA pro Sekunde auszuführen. Diese Diskussion wird im Ausblick der vorliegenden Arbeit weitergeführt.

[12] Das Problem wurde mit dem Programm *GENSRC* (Repetto 2004) erstellt (Problemklasse *a*, Instanz *a*).
[13] Das Problem wurde mit der Software *TSP Solver* erstellt (Miatidis 2004).
[14] Problem entstammt der *TSPLIB* (Reinelt 1991).

Zusammenfassung und Ausblick

Zusammenfassung

Diese Dissertation stellt ein konstruktives dynamisches System vor. Kennzeichen eines solchen Systems ist die Bildung neuer Komponenten, die selbst wiederum auf die Systemdynamik einwirken. Eine Modellierung kann im Rahmen einer Künstlichen Chemie (KC) erfolgen, ebenso eine Simulation und Analyse. Diese an die Chemie angelehnten, aber abstrakteren Modelle sind sehr gut geeignet, hochgradig parallele Systeme zu simulieren. Das globale Verhalten ergibt sich emergent aus dem Zusammenspiel einer Vielzahl von lokalen Interaktionen, die Reaktionen genannt werden. Diese sind meist elementarer Natur und finden zwischen wenigen Molekülen (üblicherweise zwei oder drei) statt. Die vorliegende Arbeit führt durch Festlegung der Objektstruktur sowie der Reaktionsvorschrift eine besondere Form der KC ein: die resolutionsbasierte Künstliche Chemie, kurz *RESAC*. Diese stellt ein spezielles System dar, das sowohl einen vereinfachten Umgang mit KC-Systemen als auch eine einheitliche Konstruktion und Analyse ermöglicht. Da das Reaktionsverhalten implizit gegeben, aber prinzipiell bekannt ist, bleibt dem Anwender allein die Aufgabe, die Moleküle zu definieren. Deren Struktur wiederum ist vorgegeben, und zwar durch die Syntax der Prädikatenlogik 1. Ordnung. Bei dieser Art der Programmierung ist kein tief gehendes Wissen über die Theorie Künstlicher Chemien nötig. Um vom verteilten Rechnen ohne Steuerinstanz, ferner von der Parallelität und von Emergenzphänomenen zu profitieren, muss allein eine auf der Syntax der Prädikatenlogik aufbauende Problemrepräsentation gefunden werden. Möglicherweise öffnet dies die KC-Forschung für einen größeren Anwenderkreis, vielleicht auch für die Industrie.

Mit dem System *RESAC* wurde außer seiner Konstruktion auch eine Parallelisierung realisiert. Diese ermöglicht einerseits ein Rechnen mit einem Mehrprozessorsystem, das wie das Internet auch verteilt organisiert sein kann. Andererseits können sich die parallel ablaufenden Prozesse auch einen Prozessor teilen. Vor dem Hintergrund des neu entwickelten Systems wurden Aspekte allgemeiner Künstlicher Chemien untersucht. Insbesondere findet eine Analyse der Ersetzungsmethodik statt, die festlegt, wie mit einem Reaktionsprodukt verfahren wird. Es geht dabei um die Frage, ob das Produkt einer Reaktion ein beliebiges Molekül oder ein an der Reaktion beteiligtes ersetzt. Eine andere Frage ist, welche Auswirkung die Einschränkung der Reaktionsregelmenge auf die möglichen Anwendungsgebiete hat. Es zeigt sich, dass *RESAC* aufgrund der strukturellen Eigenschaften der Resolutionsregel nicht für beliebige Modellierungen geeignet ist, wie grundlagentheoretische Experimente aus dem Bereich der Evolutionstheorie zeigen. Gut einsetzbar ist *RESAC* jedoch auf dem Gebiet der anwendungsorientierten Problemlösung. Das System hat hier mindestens die gleiche Ausdruckskraft wie herkömmliche Programmiersprachen.

Ein großer Anwendungsbereich für die resolutionsbasierte Künstliche Chemie ist das automatische Theorembeweisen. In dieser Arbeit wurde gezeigt, wie mit *RESAC* Beweise geführt werden können. Die hohe Ausdruckskraft der zugrunde liegenden Prädikatenlogik 1. Ordnung ist hier von zentraler Bedeutung. Diese ermöglicht unter Anderem auch eine Programmierung der KC. Eine Möglichkeit dazu ist die direkte Umsetzung der Aufgabenstellung in die Logik. Ein Beweis gleicht dann der Lösung. Im Vergleich zur logischen Programmierung befolgt die Künstliche Chemie *RESAC* ein gänzlich anderes Paradigma. Eine Lösung ist ein emergenter Prozess, der auf einer großen Anzahl parallel ausgeführter lokaler Einzelschrittberechnung beruht. Jede dieser Berechnungen ist von elementarer Natur. Eine zweite Möglichkeit, eine Aufgabenstellung in *RESAC* zu formulieren und so eine KC zu programmieren, ist das Design künstlicher Eingabemoleküle, die zwar wiederum prädikatenlogische Struktur haben, jedoch nicht mehr über die mit der Prädikatenlogik assoziierte Semantik verfügen. Die Programmierung eines Entscheidungs- und eines Optimierungsproblems lässt diesen Prozess erkennen. Beide Probleme gehören in der Informatik zu den schwierigsten unter den Problemen, die man gegenwärtig für nicht effizient lösbar hält.

Sehr interessant ist der Ausblick auf Wirkungen und Ergebnisse, die aus der Übertragung auf natürliche Prozesse mit inhärentem massiven Parallelismus resultieren. Es sei angemerkt, dass umgekehrt auch die Übertragung von Prinzipien aus dem Bereich der molekularen Prozesse auf die KC-Forschung gewinnbringend sein kann. So nutzt Banzhaf (1995) eine Eigenart der Interaktion zwischen RNA-Strängen[1] zum Aufbau einer konstruktiven Künstlichen Chemie, die das Studium von Selbstorganisation und Metabolismus ermöglicht, insbesondere die Untersuchung der Dynamik von Wettbewerb und Kooperation.

Ausblick

Hohe Informationsdichte und Reaktionsgeschwindigkeit, ferner ein hoher Grad an Parallelität sind einige der Vorzüge natürlicher Prozesse. Eine Übertragung von *RESAC* in solch ein natürliches Berechnungssystem wäre daher überaus nützlich. Wie in den bahnbrechenden Adleman-Experimenten (1994) könnte die Berechnung schwierigster Probleme auf ein molekulares System übertragen werden. Zusätzlich aber wäre ein problemunspezifischer Ansatz verwirklicht, mit dem unterschiedliche Probleme durch einen festgelegten Konstruktionsvorgang der Eingabemoleküle, nämlich durch eine Umwandlung in prädikatenlogische Strukturen, auf molekularer Ebene parallel gelöst werden könnten.

Aus der Molekularbiologie bieten sich verschiedene organische oder nicht-organische Moleküle an. Ein Biomolekül, das der Baustein künftiger Computergenerationen sein könnte, ist Bacteriorhodopsin. Dieses Molekül findet sich in der Membran von *Halobacterium halobium*. Es gehört zu einem natürlichen Farbstoff, der das Licht der Sonne aufnimmt und mit Hilfe der Photosynthese verwertet. Bacteriorhodopsin erfährt bei Bestrahlung mit Licht eine Folge struktureller Änderungen, die zum Verschlüsseln von Informationen genutzt werden können.[2]

[1] Ribonukleinsäure, abgekürzt RNS. Es wird hier die gebräuchlichere englische Abkürzung RNA verwendet.

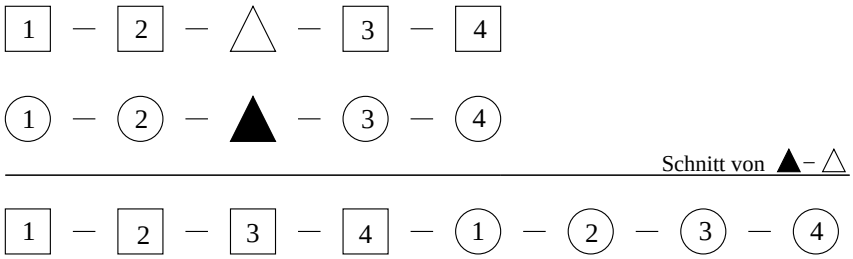
[2] Eine aufschlussreiche Beschreibung von Bakteriorhodopsin gibt Borchard-Tuch (2000) im Kapitel *Bacteriorhodopsin – der andere Stoff, aus dem Computer sein werden?*

Im Folgenden soll auf den DNA-Computer eingegangen werden, den wohl bekanntesten Vertreter der Biomolekülrechner. Die DNA[3] besteht aus langen Ketten von komplexen Molekülen. Jede DNA-Kette kodiert mit ihren Molekülen Information. Enzyme können diese Ketten an vorbestimmten Stellen teilen oder auch wieder zusammensetzen. Komplementäre Ketten können zu einem Doppelstrang hybridisieren. Biochemische Operationen synthetisieren, extrahieren, modifizieren und vervielfältigen DNA-Ketten. Ein Vorteil der DNA-Computer ist die große Informationsdichte (ein Kubikzentimeter DNA kann 10^{21} Bits Information speichern). Alle Operationen laufen parallel ab, so dass in einem Reagenzglas etwa 10 Billionen Berechnungen gleichzeitig ablaufen können. Der Energieaufwand pro Operation liegt dabei um etwa 10 Größenordnungen unter dem heutiger Supercomputer.

Eine Übertragung von *RESAC* auf DNA-Moleküle erfordert grundsätzlich zwei Überlegungen. Wie können Klauseln durch DNA-Ketten, auch Sequenzen genannt, kodiert werden? Ferner: wie kann die Inferenzregel Resolution durch biochemische Operationen nachgebildet werden?

Techniken zur Wissensrepräsentation durch DNA-Oligonukleotide wurden schon entwickelt. Ein Ziel dabei ist, DNA-Sequenzen mit günstigen Eigenschaften herzustellen. Sie müssen insbesondere tolerant gegenüber fehleranfälligen biochemischen Operationen sein. Feldkamp et al. (2000), Kim et al. (2002) sowie Feldkamp et al. (2003) stellten hierzu DNA-Sequenzcompiler her.

Eine funktionsgerechte Nachbildung der Inferenzregel ist dagegen schwieriger. Grundsätzlich erscheint die Resolutionsregel geeignet für eine Ersetzung durch biochemische Operationen. Das folgende Schema stellt den Schnitt komplementärer unifizierbarer Literale als Teil ihrer Funktionsweise dar.



Aus DNA-Perspektive könnte die Resolutionsregel wie folgt beschrieben werden: Komplementäre Strukturen lagern sich an (Hybridisierung). An klar definierten Stellen werden die Sequenzen aufgebrochen (Digestion) und unter Auslassung der hybridisierten Sequenzen wieder zusammengefügt (Ligation). Die übrigen Teile lösen sich auf. Die oben stehende Darstellung deckt allerdings nur einen Teil der Resolutionsregel ab. Nicht angeführt wurde die Unifikation und die Substitution, die die Instanziierungskomponente der Resolution ausmachen. Beispielsweise resolvieren nicht nur die Literale $P(a)$ und $\overline{P(a)}$, sondern auch $Q(f(X),Y)$ und $\overline{Q(Z,f(a))}$. Die Substitutionen $Z/f(X)$ und $Y/f(a)$ müssen anschließend auf das gesamte Ergebnismolekül angewendet werden.

Zum Ende dieses Ausblicks werden zwei Arbeiten genannt, die die enormen Fortschritte der DNA-Computer-Forschung dokumentieren und damit die oben wiedergegebenen Ideen rechtfertigen. Die erste Arbeit stammt von Wasiewicz et al. (1999) und zeigt die Realisierung eines Inferenzschrittes durch DNA-Moleküle. Es handelt sich hierbei um Prämissen-Konklusionsregeln

[3] Desoxyribonukleinsäure, abgekürzt DNS. Es wird hier die gebräuchlichere englische Abkürzung DNA verwendet.

(IF-THEN-Regeln). Die folgende Abbildung zeigt, dass dabei ein freies Ende von teilweise hybridisierten DNA-Sequenzen die Prämisse kodiert, während sich das andere freie Ende des komplementären Strangs als Konklusion zur Bindung anbietet.

Wie in *RESAC* auch wird bei diesem Ansatz die Vorwärts- und Rückwärtsverkettung unterstützt. Die die Regeln darstellenden DNA-Sequenzen können mit anderen Regeln oder mit DNA-Einzelsträngen hybridisieren. Einzelstränge repräsentieren dabei die Fakten der Problemstellung.

Die zweite Arbeit stammt von Kobayashi et al. (1997) und zeigt eine andere interessante DNA-Realisierung eines Kalküls. Die Autoren präsentieren eine Implementierung einer eingeschränkten Hornlogik, in der keine Prädikate erlaubt sind. Sei $V = \{A, B, C, \ldots\}$ eine abzählbare Menge Boolescher Variablen. Dargestellt werden können Formeln der Form $A \wedge B \rightarrow C$ oder A, wobei die unverknüpfte Variable ein Faktum repräsentiert. Umgangen bzw. ausgelassen ist hier der Unifikations- und Substitutionsprozess.

Eine DNA-Implementierung des Unifikations- und Substitutionsprozesses wird die entscheidende Aufgabe im Übertragungsprozess der resolutionsbasierten Künstlichen Chemie sein. Die Realisierung ist ein zurzeit noch ungelöstes Problem.

Jens Busch

im April 2004.

Über den Autor

Von 1993 bis 1998 Studium der Informatik an der Universität Dortmund mit Nebenfach Theoretische Medizin an der Ruhr-Universität Bochum, Abschluss Diplom-Informatiker. Von 1999 bis 2003 wissenschaftlicher Mitarbeiter in der Arbeitsgruppe von Prof. Dr. Wolfgang Banzhaf am Lehrstuhl für Systemanalyse der Universität Dortmund, Abteilung Anwendung der Informatik in den Ingenieurswissenschaften. Schwerpunkte der wissenschaftlichen Tätigkeit lagen auf den Gebieten der Künstlichen Intelligenz und der Artificial Life-Forschung, insbesondere der Künstlichen Chemie. Zu den weiteren Interessen gehört das emergente Rechnen sowie das simulierte molekulare Rechnen. Darüber hinaus Veröffentlichungen in den Bereichen Multiagentensysteme und Genetische Programmierung. Hier ist insbesondere – auf das Gebiet der Robotik übergreifend – die automatische Programmierung von Steuerungsprogrammen für beliebige Laufroboter-Architekturen zu nennen (`http://ls11-www.cs.uni-dortmund.de/people/sigel`). Im Jahre 2000 zweimonatiges Auslandsstipendium an der Seoul National University (SNU) in Südkorea in der Arbeitsgruppe von a. o. Prof. Dr. Byoung-Tak Zhang, Dept. of Engineering, Abteilung Artificial Intelligence Lab (SCAI).

Publikationen

Busch, J. (1999). Automated Theorem Proving for first order predicate calculus using Artificial Chemistries. In G. Gesellschaft für Informatik (Hg.), *Informatiktage 1999*, Bad Schussenried. Konradin Verlag Robert Kohlhammer GmbH.

Busch, J. und W. Banzhaf (2000a). Multi-Agent Systems inspired by Artificial Chemistries: A Case Study in Automated Theorem Proving. In *Proceedings, Fourth International Conference on MultiAgent Systems*, Boston, MA, USA, S. 371–372. IEEE Computer Society.

Busch, J. und W. Banzhaf (2000b). The Construction of an Automated Theorem Prover based on Emergent Systems. In K. K. Lai, O. Katai, M. Gen und B. Liu (Hg.), *Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Applications*, S. 132–143. Global-Link Publishing Company.

Busch, J. und W. Banzhaf (2003). How to Program Artificial Chemistries. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim und J. Ziegler (Hg.), *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, Volume 2801 of *Lecture Notes in Artificial Intelligence*, S. 20–30. Springer Verlag Berlin, Heidelberg.

Busch, J., W. Kantschik, H. Aburaya, K. Albrecht, R. Groß, P. Gundlach, M. Kleefeld, A. Skusa, M. Villwock, T. Vogd und W. Banzhaf (2000). Evolution von GP-Agenten mit Schachwissen sowie deren Integration in ein Computerschachsystem (PG- Endbericht). Technical Report SYS-1/00, University of Dortmund, Department of Computer Science, Systems Analysis Research Group (LS XI).

Busch, J., J. Ziegler, C. Aue, A. Ross, D. Sawitzki und W. Banzhaf (2002). Automatic Generation of Control Programs for Walking Robots Using Genetic Programming. In J. A. Foster, E. Lutton, J. F. Miller, C. Ryan und A. Tettamanzi (Hg.), *Genetic Programming, 5th European Conference, EuroGP 2002, Kinsale, Ireland, April 3-5, 2002, Proceedings*, Volume 2278 of *Lecture Notes in Computer Science*, S. 258–267. Springer.

Skusa, A., W. Banzhaf, J. Busch, P. Dittrich und J. Ziegler (2000). Künstliche Chemie. *Künstliche Intelligenz 1*, 12–19.

Ziegler, J., J. Barnholt, J. Busch und W. Banzhaf (2002). Automatic evolution of control programs for a small humanoid walking robot. In P. Bidaud und F. B. Amar (Hg.), *Proceedings of the 5th International Conference on Climbing and Walking Robots (CLAWAR)*, S. 109–116. Professional Engineering Publishing.

Danksagung

Für ihre Hilfe bei der Fertigstellung dieser Dissertation möchte ich danken:

Wolfgang Banzhaf

Eun-Ok Busch

Horst Busch

Peter Dittrich

Peter Herrmann

Niels Lepperhoff

Peter Marwedel

Heinrich Müller

Andre Skusa

Jens Ziegler

Darüber hinaus danke ich Herrn Byoung-Tak Zhang für die Aufnahme in seine Arbeitsgruppe am Artificial Intelligence Lab (SCAI) der Seoul National University (SNU) in Südkorea.

Literaturverzeichnis

Adami, C. (1995). Self-organized criticality in living systems. *Phys. Lett. A 203*, 29–32.

Adami, C. und C. T. Brown (1994). Evolutionary learning in the 2D artificial life system avida. In R. A. Brooks und P. Maes (Hg.), *Artificial Life IV*, Cambridge, MA, S. 377–381. MIT Press.

Adleman, L. M. (1994). Molecular Computation of Solutions to Combinatorial Problems. *Science 266*, 1021–1024.

Albert, L., R. Casas, F. Fages und P. Zimmermann (1991). Average case analysis of unification algorithms. In M. Jantzen (Hg.), *Proceedings 8. STACS — GI-Symposium on Theoretical Aspects of Computer Science*, Volume 480 of *Lecture Notes in Computer Science*, Berlin, S. 196–213. Springer.

Astor, J. C. und C. Adami (1998). Development of evolution of neural networks in an artificial chemistry. In C. Wilke, S. Altmeyer und T. Martinetz (Hg.), *Third German Workshop on Artificial Life*, S. 15–30. Verlag Harri Deutsch, Frankfurt a. M.

Astor, J. C. und C. Adami (2000). A developmental model for the evolution of artificial neural networks. *Artif. Life 6*(3), 189–218.

Atkins, P. W. (1994). *Physical Chemistry*. Oxford University Press.

Bäck, T., D. B. Fogel und Z. Michalewicz (Hg.) (1997). *Handbook of Evolutionary Computation*. Oxford University Press, New York, NY and Institute of Physics Publishing, Bristol.

Bagley, R. J. und J. D. Farmer (1992). Spontaneous emergence of a metabolism. In C. G. Langton, C. Taylor, J. D. Farmer und S. Rasmussen (Hg.), *Artificial Life II*, Redwood City, CA, S. 93–140. Addison-Wesley.

Banzhaf, W. (1992). Competition as an organizational principle for massively parallel computers? In *Proceedings of the Workshop on, Physics and Computation*, Dallas, S. 229 — 231. IEEE Computer Society Press, Los Alamitos.

Banzhaf, W. (1993). Self-replicating sequences of binary numbers – foundations I and II: General and strings of length n = 4. *Biol. Cybern. 69*, 269–281.

Banzhaf, W. (1994a). Self-organization in a system of binary strings. In R. Brooks und P. Maes (Hg.), *Artificial Life IV*, Cambridge, MA, S. 109–119. MIT Press.

Banzhaf, W. (1994b). Self-replicating sequences of binary numbers: The build-up of complexity. *Complex Syst. 8*, 215–225.

Banzhaf, W. (1995). Self–organizing algorithms derived from RNA interactions. In W. Banzhaf und F. Eeckman (Hg.), *Evolution and Biocomputation – Computational Models of Evolution*, Volume 899 of *Lecture Notes in Computer Science, LNCS*, S. 69–102. Berlin: Springer.

Banzhaf, W. (2002). On the Dynamics of Competition in a simple Artificial Chemistry. *Nonlinear Phenomena in Complex Systems 5*, 318 – 324.

Banzhaf, W. (2004). Artificial chemistries – Toward Constructive Dynamical Systems. *Solid State Phenomena*. Erscheint 2004.

Banzhaf, W., P. Dittrich und B. Eller (1999). Selforganization in a system of binary strings with topological interactions. *Physica D 125*, 85–104.

Banzhaf, W., P. Dittrich und H. Rauhe (1996). Emergent computation by catalytic reactions. *Nanotechnology 7*(1), 307–314.

Benenson, Y., T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh und E. Shapiro (2001). Programmable and autonomous computing machine made of biomolecules. *Nature 414*(1), 430–434.

Benkö, G., C. Flamm und P. F. Stadler (2003). Generic Properties of Chemical Networks: Artificial Chemistry Based on Graph Rewriting. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim und J. Ziegler (Hg.), *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, Volume 2801 of *Lecture Notes in Artificial Intelligence*, S. 10–19. Springer Verlag Berlin, Heidelberg.

Berry, G. und G. Boudol (1992). The chemical abstract machine. *Theoretical Computer Science 96*(1), 217–248.

Bersini, H. (2000). Reaction mechanisms in the oo chemistry. In M. A. Bedau, J. S. McCaskill, N. H. Packard und S. Rasmussen (Hg.), *Artificial Life VII*, Cambridge, MA, S. 39–48. MIT Press.

Bibel, W. (1992). *Deduktion, Automatisierung der Logik*. Oldenbourg Verlag.

Bisswanger, H. (2000). *Enzymkinetik: Theorie und Methoden*. Weinheim: WILEY-VCH Verlag.

Bläsius, K. H. und H.-J. Bürckert (1992). *Deduktionssysteme*. München: Oldenbourg Verlag. 2. Auflage.

Boerlisr, M. C. und P. Hogeweg (1991). Spiral wave structures in prebiotic evolution: Hypercycle stable against parasites. *Physica D 48*, 17–28.

Boolos, G. und R. Jeffrey (1974). *Computability and Logic*. London, UK: Cambridge University Press.

Borchard-Tuch, C. (2000). *Herausforderung: Biocomputer oder Zurück zur Natur*. Georg Olms Verlag AG.

Brachman, R. J. und H. J. Levesque (1984). The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, Austin, TX, S. 34–37.

Centler, F., P. Dittrich, L. Ku, N. Matsumaru, J. Pfaffmann und K.-P. Zauner (2003). Artificial Life as an Aid to Astrobiology: Testing Life Seeking Techniques. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim und J. Ziegler (Hg.), *Advances in Artificial Life - Proceedings*

of the 7th European Conference on Artificial Life (ECAL), Volume 2801 of *Lecture Notes in Artificial Intelligence*, S. 31–40. Springer Verlag Berlin, Heidelberg.

Chang, C.-L. und R. Lee (1973). *Symbolic Logic and Mechanical Theorem Proving*. Boston: Academic Press.

Church, A. (1936). A Note on the Entscheidungsproblem. *Journal Symbolic Logic* (1), 40–41.

Clocksin, W. F. und C. S. Mellish (1981). *Programming in Prolog*. Berlin, Heidelberg: Springer.

Denbigh, K. G. und J. Turner (1971). *Chemical Reactor Theory: An Introduction* (2. Auflage Auflage). London, UK: Cambridge University Press.

Denbigh, K. G. und J. Turner (1973). *Einführung in die chemische Reaktionstechnik*. Verlag Chemie GmbH. Übersetzung von Denbigh und Turner (1971).

Dewdney, A. K. (1984). In the Game Called Core War Hostile Programs engage in a Battle of Bits. *Sci. Amer. 250*, 14–22.

Dittrich, P. (1995). Selbstorganisation in einem System von Binärstrings mit algorithmischen Sekundärstrukturen.

Dittrich, P. (1998). Real evolution in artificial chemistries. In C. L. Nehaniv und G. P. Wagner (Hg.), *The Right Stuff: Appropriate Mathematics for Evolutionary and Developmental Biology*, S. 27–31. Technical Report, Computer Science, University of Hertfordshire, School of Information Science, No. 315.

Dittrich, P. (2001). *On Artificial Chemistries*. Dissertation, Chair of Systems Analysis, Department of Computer Science, University of Dortmund, D-44221 Dortmund, Germany.

Dittrich, P., F. Liljeros, A. Soulier und W. Banzhaf (2000). Spontaneous Group Formation in the Seceder Model. *Phy. Rev. Lett 84*, 3205–3208.

Dittrich, P., J. Ziegler und W. Banzhaf (2001). Artificial Chemistries - a Review. *Artificial Life 7*(3), 225–275.

Doyle, J. und R. S. Patil (1991). Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence 48*(3), 261–298.

Ebbinghaus, H.-D., J. Flum und W. Thomas (1978). *Einführung in die mathematische Logik*. Darmstadt: Wissenschaftliche Buchgesellschaft.

Eigen, M. (1971). Selforganisation of matter and the evolution of biological macromolecules. *Naturwissenschaften 58*, 465–526.

Eigen, M. (2002). Angst vor Viren. *TK aktuell spezial - Das Magazin der Techniker Krankenkasse*. Interview mit Manfred Eigen.

Eigen, M., W. Gardiner, P. Schuster und R. Winkler-Oswatitsch (1981). Ursprung genetischer Information. *Spektrum der Wissenschaft* (6), 36–56.

Eigen, M. und P. Schuster (1977). The Hypercycle: a Principle of Natural Self-Organisation, Part A. *Naturwissenschaften 64*(11), 541–565.

Eigen, M. und P. Schuster (1979). *The Hypercycle*. Berlin: Springer.

Farmer, J. D., S. A. Kauffman und N. H. Packard (1986). Autocatalytic replication of polymers. *Physica D 22*, 50–67.

FAZ (2004). Frankfurter Allgemeine Zeitung. 14.01.2004, Nr. 11 / Seite N2, Text: mli.

Feldkamp, U., W. Banzhaf und H. Rauhe (2000). A DNA sequence compiler. In A. Condon und G. Rozenberg (Hg.), *Preliminary Proc. Sixth DIMACS Workshop on DNA-Computing*, S. 253. Leiden, The Netherlands.

Feldkamp, U., H. Rauhe und W. Banzhaf (2003). Software tools for DNA sequence design. *Genetic Programming and Evolvable Machines 4*(2), 153–171.

Fitting, M. (1996). *First-Order Logic and Automated Theorem Proving*. New York: Springer-Verlag. 2. Auflage.

Fontana, W. (1992). Algorithmic chemistry. In C. G. Langton, C. Taylor, J. D. Farmer und S. Rasmussen (Hg.), *Artificial Life II*, Redwood City, CA, S. 159–210. Addison-Wesley.

Fontana, W. und L. W. Buss (1994). 'The arrival of the fittest': Toward a theory of biological organization. *Bull. Math. Biol. 56*, 1–64.

Fontana, W. und L. W. Buss (1996). The barrier of objects: From dynamical systems to bounded organization. In J. Casti und A. Karlqvist (Hg.), *Boundaries and Barriers*, Redwood City, MA, S. 56–116. Addison-Wesley.

Fontana, W., G. Wagner und L. W. Buss (1994). Beyond digital naturalism. *Arficial Life 1/2*, 211–227.

Frege, G. (1882). Über den Zweck der Begriffsschrift. In *Gottlob Frege, Begriffsschrift*. Hildesheim: G. Olms Verlagsbuchhandlung, 1964.

Furusawa, C. und K. Kaneko (1998). Emergence of Rules in Cell Society: Differentiation, Hierarchy, and Stability. *Bull. Math. Bio. 60*, 659–87.

Furusawa, C. und K. Kaneko (2000). Complex organizations in multicellularity as a necessity in evolution. In M. A. Bedau, J. S. McCaskill, N. H. Packard und S. Rasmussen (Hg.), *Artificial Life VII*, Cambridge, MA, S. 103–12. MIT Press.

Gallaire, H., J. Minker und J.-M. Nicolas (1984). Logic and Databases: A Deductive Approach. *ACM Comput. Surv. 16*(2), 153–185.

Garzon, M. H. und R. J. Deaton (1999). Biomolecular computing and programming. *IEEE Trans. on Evolutionary Computation 3*(3), 236–250.

Genesereth, M. R. und N. Nilsson (1987). *Logical Foundations of Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann Publishers.

Genesereth, M. R. und N. Nilsson (1989). *Logische Grundlagen der Künstlichen Intelligenz*. Braunschweig: Friedrich Vieweg & Sohn.

Gentzen, G. (1935). Untersuchungen über das logische Schließen. *Mathematische Zeitschrift 39*, 176–210.

Gini, M. (1995). Lecture notes. Internet: `http://www-users.cs.umn.edu/~gini/5511/logic`. Letzter Zugriff: Oktober 1999.

Goubault-Larrecq, J. und I. Mackie (1997). *Proof Theory and Automated Deduction*, Volume 6 of *Applied Logic Series*. Dordrecht: Kluwer Academic Publishers.

Green, C. (1969). Theorem-Proving by Resolution as a Basis for Question-Answering Systems. In B. Meltzer und D. Michie (Hg.), *Machine Intelligence 4*, S. 183–205. Edinburgh: Edinburgh University Press.

Grolmusz, V. (1991). Large parallel machines can be extremely slow for small problems. *Algorithmica 6*(4), 479–489.

Grosche, G., V. Ziegler, D. Ziegler und E. Zeidler (Hg.) (1995). *Teubner-Taschenbuch der Mathematik, Teil II*. Stuttgart: B. G. Teubner. 7. Auflage.

Gross, R., K. Albrecht, W. Kantschik und W. Banzhaf (2002). Evolving chess playing programs. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke und N. Jonoska (Hg.), *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, New York, S. 740–747. Morgan Kaufmann Publishers.

Grötschel, M. und M. W. Padberg (1993). Ulysses 2000: In search of optimal solutions to hard combinatorial problems. Technical Report 93-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin.

Guldberg, C. M. und P. Waage (1879). Concerning chemical affinity. *J. prakt. Chem. 19*(2), 69–114.

Gödel, K. (1930). Die Vollständigkeit der Aximome des Logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik 37*, 349–360.

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik 38*, 173–198.

Görtz, G., C.-R. Rollinger und J. Schneeberger (Hg.) (2000). *Handbuch der Künstlichen Intelligenz*. München: Oldenbourg Verlag. 3. Auflage.

Haken, A. (1985). The intractability of resolution. *Theoretical Computer Science 39*, 297–308.

Hales, T. C. (2002). A computer verification of the Kepler conjecture. *Proceedings of the ICM 3*, 795–804.

Hayes, P. J. (1973). Computation and deduction. In *Proceedings of the 1973 Mathematical Foundations of Computer Science Symposium*, Czechoslovakian Academy of Sciences.

Herbrand, J. J. (1930). *Recherches sur la théorie de la démonstration*. Dissertation, University of Paris. Ausschnitte ins Englische übersetzt von (van Heijenoort 1967).

Herrmann, P. und H. Krumm (2000). A framework for modeling transfer protocols. *Computer Networks 34*, 317–337.

Hill, C. G. (1977). *An introduction to chemical engineering kinetics and reactor design*. New York: John Wiley & Sons.

Hofbauer, D. und R.-D. Kutsche (1991). *Grundlagen des maschinellen Beweisens*. Braunschweig: Friedrich Vieweg & Sohn. 2. Auflage.

Hofbauer, J. und K. Sigmund (1984). *Evolutionstheorie und dynamische Systeme*. Paul Parey, Berlin-Hamburg.

Hofbauer, J. und K. Sigmund (1988). *Dynamical Systems and the Theory of Evolution*. University Press, Cambridge UK.

Hofbauer, J. und K. Sigmund (1998). *Evolutionary Games and Population Dynamics*. Cambridge, UK: Cambridge University Press.

Huet, G. (1987). Deduction and Computation. In W. Bibel und P. Jorrand (Hg.), *Fundamentals of Artificial Intelligence – An Advanced Course*, Volume 232 of *Lecture Notes in Computer Science*, S. 39–74. Berlin: Springer.

Hutton, T. J. (2002). Evolvable self-replicating molecules in an artificial chemistry. *Artificial Life 8*(4), 341–356.

Hutton, T. J. (2003). Simulating Evolution's First Steps. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim und J. Ziegler (Hg.), *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, Volume 2801 of *Lecture Notes in Artificial Intelligence*, S. 51–58. Springer Verlag Berlin, Heidelberg.

Ikegami, T. und T. Hashimoto (1995). Active mutation in self-reproducing networks of machines and tapes. *Artificial Life 2*(3), 305–318.

JaJa, J. (1992). *An Introduction to Parallel Algorithms*. Addison-Wesley.

Johnstone, P. T. (1992). *Notes on Logic and Set Theory*. Cambridge University Press.

Kampis, G. (1991). *Self-modifying systems in biology and cognitive science*. Oxford, UK: Pergamon Press.

Kanada, Y. (1995a). Combinatorial problem solving using randomized dynamic composition of production. In *Proceedings of the Second IEEE Conference on Evolutionary Computation*, S. 467–472.

Kanada, Y. (1995b). Combinatorial problem solving using randomized dynamic tunneling on a production system. In *IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century '95*, Volume 4, New York, NY, S. 3784–9. IEEE Computer Society.

Kanada, Y. und M. Hirokawa (1994). Stochastic Problem Solving by Local Computation based on Self-organization Paradigm. In *27th Hawaii International Conference on System Science*, S. 82–91. IEEE Computer Society.

Kauffman, S. A. (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. New York: Oxford University Press.

Kim, D., S.-Y. Shin, I.-H. Lee und B.-T. Zhang (2002). NACST/Seq: A Sequence Design System with Multiobjective Optimization. In *Proc. 8th International Workshop on DNA Based Computers*, Volume 2568 of *Lecture Notes in Computer Science*, S. 242–251. Springer Verlag.

Knight, K. (1989). Unification: A multidisciplinary survey. *ACM Computing Surveys 21*, 93–124.

Kobayashi, S., T. Yokomori, G. ichi Sampei und K. Mizobuchi (1997). DNA Implementation of Simple Horn Clause Computation. In *Proc. of IEEE International Conference on Evolutionary Computation'97*, S. 213–217.

Koudriavtsev, A. B., R. F. Jameson und W. Linert (2001). *The Law of Mass Action*. Springer Verlag.

Kowalski, R. (1974). Predicate logic as a programming language. In *Information Processing 74*, S. 569–574. North-Holland.

Kowalski, R. (1979a). Algorithm = logic + control. *Communications of the ACM 22*(7), 424–436.

Kowalski, R. (1979b). *Logic for Problem Solving*. New York, NY: Elsevier North-Holland.

Küppers, B. O. (1983). *Molecular theory of evolution*. Berlin: Springer-Verlag.

Lenski, R. E., C. Ofria, T. C. Collier und C. Adami (1999). Genome complexity, robustness and genetic interactions in digital organisms. *Nature 400*(6745), 661–4.

Levesque, H. J. (1986). Making believers out of computers. *Artificial Intelligence 30*(1), 81–108.

Lindström, P. (1969). On extensions of elementary logic. *Theoria 35*.

Loveland, D. W. (1978). *Automated Theorem Proving: A Logical Basis*. New York: North-Holland.

Luckham, D. C. und N. Nilsson (1971). Extracting Information from Resolution Proof Trees. *Artificial Intelligence 2*(1), 27–54.

Luisi, P. L. (1991). The chemical implementation of autopoisis. In G. R. Fleischaker, S. Colonna und P. L. Luisi (Hg.), *Self-Production of Supramolecular Structures*, Dordrecht, S. 179–197. Kluwer.

M. Hall, J. (1968). *The Theory of Groups*. The Macmillan Company.

Madina, D., N. Ono und T. Ikegami (2003). Cellular Evolution in a 3D Lattice Artificial Chemistry. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim und J. Ziegler (Hg.), *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, Volume 2801 of *Lecture Notes in Artificial Intelligence*, S. 59–68. Springer Verlag Berlin, Heidelberg.

Martelli, A. und U. Montanari (1982). An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems 4*(2), 258–282.

Mayer, B. und S. Rasmussen (1998). Self-reproduction of dynamical hierachies in chemical systems. In C. Adami, R. Belew, H. Kitano und C. Taylor (Hg.), *Artificial Life VI*, Cambridge, MA, S. 123–129. MIT Press.

McCaskill, J. S. (1988). Polymer chemistry on tape: A computational model for emergent genetics. Internal report, MPI for Biophysical Chemistry, Göttingen, Germany.

McCaskill, J. S., H. Chorongiewski, D. Mekelburg, U. Tangen und U. Gemm (1994). Configurable computer hardware to simulate long-time self-organization of biopolymers. *Ber. Bunsenges. Phys. Chem. 98*(9), 1114–1114.

McCune, W. (1990). OTTER 2.0. In M. E. Stickel (Hg.), *Proceedings of the 10th International Conference on Automated Deduction*, Volume 449 of *Lecture Notes in Artificial Intelligence*, Kaiserslautern, Germany, S. 663–664. Springer Verlag.

McMullin, B. und F. J. Varela (1997). Rediscovering computational autopoiesis. In P. Husbands und I. Harvey (Hg.), *Fourth European Conference on Artificial Life*, Cambridge, MA, S. 38–47. MIT Press.

Miatidis, M. (2004). Software *TSP Solver*, Version: 2.0. `http://kickme.to/GrMikeD`. Letzter Zugriff: 05.02.2004.

Nebel, B. (1995). Komplexitätsanalysen in der Künstlichen Intelligenz. *Künstliche Intelligenz 95/2*.

Nerode, A. und R. A. Shore (1997). *Logic for Applications*. New York: Springer-Verlag. 2. Auflage.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann.

Nilsson, N. J. (1998). *Artificial Intelligence: A new Synthesis*. Morgan Kaufmann.

Nordin, P. und W. Banzhaf (1997). Genetic reasoning evolving proofs with genetic search. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba und R. L. Riolo (Hg.), *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, USA, S. 255–260. Morgan Kaufmann.

Ono, N. und T. Ikegami (1999). Model of self-replicating cell capable of self-maintenance. In D. Floreano, J.-D. Nicoud und F. Mondana (Hg.), *Proc. Fifth European Conference on Artificial Life (ECAL'99)*, Berlin, S. 399–406. Springer.

Ono, N. und T. Ikegami (2000). Self-maintenance and self-reproduction in an abstract cell model. *J. Theor. Biol. 206*(2), 243–253.

Ono, N. und T. Ikegami (2001). Artificial Chemistry: Computational Studies on the Emergence of Self-Reproducing Units. In J. Kelemen und P. Sosik (Hg.), *Advances in Artificial Life*, Volume 2159 of *Lecture Notes in in Computer Science*, Berlin, S. 186–195. Springer.

Pargellis, A. N. (1996). The evolution of self-replicating computer organisms. *Physica D 98*(1), 111–127.

Paterson, M. und M. Wegman (1978). Linear Unification. *Journal of Computer and System Science 16*, 158–167.

Peano, G. (1889). *Arithmetices principia, novo methodo exposita*. Turin: Fratres Bocca.

Plaisted, D. A. und Y. Zhu (1999). *The Efficiency of Theorem Proving Strategies: A Comparative and Asymptotic Analysis*. Braunschweig: Friedrich Vieweg & Sohn. 2. Auflage.

Rasmussen, S., C. Knudsen und R. Feldberg (1992). Dynamics of programmable matter. In C. G. Langton, C. Taylor, J. D. Farmer und S. Rasmussen (Hg.), *Artificial Life II*, Redwood City, CA, S. 211–291. Addison-Wesley.

Rasmussen, S., C. Knudsen, R. Feldberg und M. Hindsholm (1990). The Coreworld: Emergence and Evolution of Cooperative Structures in a Computational Chemistry. *Physica D 42*, 111–134.

Ray, T. S. (1991). *Evolution and optimization of digital organisms.* School of Life and Health Sciences, University of Delaware, Newark, Delaware 19716, USA.

Ray, T. S. (1992). An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer und S. Rasmussen (Hg.), *Artificial Life II*, Redwood City, CA, S. 371–408. Addison-Wesley.

Reinelt, G. (1991). TSPLIB— a traveling salesman problem library. *ORSA Journal on Computing 3*(4), 376–384.

Repetto, B. (2004). Software *gensrc*. `http://www.andrew.cmu.edu/~neils/testdata/GENSRC/`. Letzter Zugriff: 05.02.2004.

Robinson, G. und L. Wos (1969). Paramodulation and theorem-proving in first-order theories with equality. In B. Meltzer und D. Michie (Hg.), *Machine Intelligence 4*, S. 135–150. Edinburgh University Press.

Robinson, J. (1965, January). A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery 12*, 23–41. Nachgedruckt in (Siekmann und Wrightson 1983).

Robinson, R. M. (1950). An Essitially Undecidable Axiom System. In *Proceedings of the international Congress of Mathematics*, S. 729–730.

Rosen, R. (1991). *Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life*. New York, NY: Columbia University Press.

Russell, S. J. und P. Norvig (2003). *Artificial Intelligence: A Modern Approach*. Upper Saddle River, New Jersey: Prentice-Hall. 2. Auflage.

Schuster, P. und K. Sigmund (1983). Replicator dynamics. *J. Theor. Biol. 100*, 533–8.

Schuster, P., K. Sigmund und R. Wolff (1979). Dynamical Systems under Constant Organization III: Cooperative and Competitive Behaviour of Hypercycles. *J.Diff.Equs. 32*, 357–368.

Siekmann, J. und G. Wrightson (Hg.) (1983). *Automation of Reasoning: 1957-1970*. Berlin: Springer-Verlag. 2 Bände.

Skolem, T. (1920). Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze. *Skrifter utgit av Videnskapselskapet i Kristiana 4*, 4–36.

Skusa, A., W. Banzhaf, J. Busch, P. Dittrich und J. Ziegler (2000). Künstliche Chemie. *Künstliche Intelligenz 1*, 12–19.

Speroni di Fenizio, P. (2000). A less abstract artficial chemistry. In M. A. Bedau, J. S. McCaskill, N. H. Packard und S. Rasmussen (Hg.), *Artificial Life VII*, Cambridge, MA, S. 49–53. MIT Press.

Spiegelman, S. (1971). An approach to the experimental analysis of precellular evolution. *Quart. Rev. Biophysics 4*(2), 213–253.

Stanford Encyclopedia 2004. Stanford Encyclopedia of Philosophy: Automated Reasoning. Freie Enzyklopädie im Internet: `http://plato.stanford.edu`. letzter Zugriff: Februar 2004.

Steels, L. (1994). The artificial life roots of artificial intelligence. *Artificial Life 1*(1–2), 89–125.

Stickel, M. E. (1986). Schubert's steamroller problem: formulations and solutions. *J. Autom. Reason. 2*(1), 89–101.

Suzuki, Y., J. Takabayashi und H. Tanaka (2000). Investigation of an Ecological System by using an Abstract Rewriting System on Multisets. In G. Paun (Hg.), *Recent Topics in Mathematical and Computational Linguistics*, Bucharest, S. 300–309. The Publ. House of the Romanian Academy.

Suzuki, Y., S. Tsumoto und H. Tanaka (1996). Analysis of cycles in symbolic chemical system based on abstract rewriting system on multisets. In C. G. Langton und K. Shimohara (Hg.), *Artificial Life V*, Cambridge, MA, S. 521–528. MIT Press.

Szuba, T. (1997). Evaluation measures for the collective intelligence of closed social structures. In *International Conference on Intelligent Systems and Semiotics (ISAS'97)*, Gaithersburg, MD, USA, S. 401–406.

Szuba, T. und R. Straš (1997). Parallel evolutionary computing with the random prolog processor. *J. Parallel Distrib. Comput. 47*(1), 78–85.

Taylor, P. D. und L. Jonker (1978). Evolutionary stable strategies and game dynamics. *Math. Biosci. 40*, 154–56.

Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. In *Proceedings of the London Mathematical Society, second series*, Volume 42, S. 230–265.

van Heijenoort, J. (Hg.) (1967). *From Frege to Gödel*. Cambridge, Mass.: Harvard University Press.

Varela, F. J., H. R. Maturana und R. Uribe (1974). Autopoiesis: The organization of living systems. *BioSystems 5*(4), 187–196.

Wasiewicz, P., T. Janczak, J. Mulawka und A. Plucienniczak (1999). The inference via DNA computing. In *Proc. Congress on Evolutionary Computation (CEC'99)*, S. 988–993.

Westerterp, K. R., W. P. M. V. Swaaij und A. A. C. M. Beenackers (1987). *Chemical Reactor Design and Operation* (2. Auflage Auflage). John Wiley & Sons.

Wikipedia 2004. Wikipedia: First-order predicate calculus. Freie Enzyklopädie im Internet: `http://www.wikipedia.org`. letzter Zugriff: Februar 2004.

Winston, P. H. (1984). *Artificial Intelligence*. Reading, Mass.: Addison-Wesley Publishing Company.

Winston, P. H. (1987). *Künstliche Intelligenz*. Bonn: Addison-Wesley. Dt. Übersetzung von Peter Hamm.

Wos, L., D. Carson und G. Robinson (1965). Efficiency and completeness of the set of support strategy in theorem proving. *Journal of the Association for Computing Machinery 12*, 536–541.

Wos, L., R. Overbeek, E. Lusk und J. Boyle (1984). *Automated Reasoning: Introduction and Applications*. Prentice-Hall.

Zauner, K.-P. und M. Conrad (1998). Conformation-driven computing: Simulating the context-conformation-action loop. *Supramolecular Science 5*(5-6), 791–794.

Zhirnov, V., R. Cavin, J. Hutchby und G. Bourianoff (2003). Limits to Binary Logic Switch Scaling-A Gedanken Model. In *Proceedings of the IEEE*.

Zhou, S. (1992). LSF: load sharing in large-scale heterogeneous distributed systems. In *Proceedings of the Workshop on Cluster Computing*, Orlando, FL.

Ziegler, J. und W. Banzhaf (2000). Evolving a "nose" for a robot. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. Morgan Kaufmann.

Teil IV

Anhänge

Anhang A

Verdeutlichung des Resolutionsablaufs beim TSP

Schablonen-Endersetzung, Nicht-Mehrfacheinbindung, Protokollführung

Beispiel:

Gegeben sei ein TSP mit 5 Städten, bezeichnet durch a, b, c, d und e. Das Problem wird durch Tri-Sektionsklauseln repräsentiert. Unterteilt wird in die Sektionen *Lösungsfindung,Kosten* und *Stadtintegration* (siehe Abb. 8.15, Seite 156). Die Betrachtung im Kapitel 8.3 wird durch *Trace*-Literale erweitert, die der Protokollierung der Lösung dienen. Die Sektion *Lösungsfindung* besteht aus *Path*-Literalen, in der Sektion *Kosten* befinden sich die *Costs*-Literale. Den Teil *Stadtintegration* teilen sich die Literale *Visited* und *Trace*. Im vorliegenden Beispiel werden die folgenden 5 Stadtverbindungen (Kanten) betrachtet:

$$(a,d),(c,a),(b,c),(d,e),(e,b)$$

Die Kanten (a,d) und (c,a) werden kodiert durch die Kantenklauseln:

$\text{Path}(\text{edge}(a,d))$,

$\overline{\text{Costs}(X,Y_1)},\ldots,\overline{\text{Costs}(Y_{k-1},Y_k)},\text{Costs}(X,Y_k)$

$\overline{\text{Visited}(a)},\overline{\text{Visited}(d)},\ \text{Trace}(\text{edge}(a,d))$.

und

$\text{Path}(\text{edge}(c,a))$,

$\overline{\text{Costs}(X,Y_1)},\ldots,\overline{\text{Costs}(Y_{l-1},Y_l)},\text{Costs}(X,Y_l)$

$\overline{\text{Visited}(c)},\overline{\text{Visited}(a)},\ \text{Trace}(\text{edge}(c,a))$.

Die Kostensektion ist hier nur angedeutet. Die anderen Kantenklauseln werden analog gebildet.

Neben den Kantenklauseln gehören die Kettenschablonen$^{(5)}$ zur Menge *Startklauseln*, die die Füllung des Reaktors zu Beginn des Experiments und den Zufluss bestimmt. Wenn Start und Ziel der Rundreise o. B. d. A. in Stadt a erfolgt, haben sie den folgenden Aufbau:

$\overline{\text{Path}(\text{edge}(a,X_2))}, \overline{\text{Path}(\text{edge}(X_2,X_3))}, \overline{\text{Path}(\text{edge}(X_3,X_4))}, \overline{\text{Path}(\text{edge}(X_4,X_5))}, \overline{\text{Path}(\text{edge}(X_5,a))}$,

$\overline{\text{Costs}(X,Y)}, \text{Costs}(X,Y)$, $\text{Visited}(b), \text{Visited}(c), \text{Visited}(d), \text{Visited}(e)$

Auf die Klauseln im Reaktor wird bei Kollision die separate Multiresolution gemäß Def. 8.2.1 (Seite 145) angewendet. Sie erfolgt auf den drei Bereichen *Lösungsfindung*, *Kosten* und *Stadtintegration*, wobei Sektion *Lösungsfindung* priorisiert ist, formal notiert als $sMR_{\underline{\{\text{Path}\}},\{\text{Costs}\},\{\text{Visited, Trace}\}}$.

Es wird die Entwicklung einer Kettenschablone bei Schablonen-Endersetzung bezüglich der *Path*-Literale demonstriert. Die Entwicklung wird von der Initialisierung (Freiheitsgrad 5) bis zur Lösung (Freiheitsgrad 0) verfolgt, dabei deuten im Folgenden die unterstrichenen Literale in der *Lösungsfindung*-Sektion die reaktiven Enden der *Path*-Literale an. Die *Costs*-Literale der Kettenschablone$^{(i)}$ werden abkürzend durch $COSTS_i$ ersetzt.

Trifft Kettenschablone$^{(5)}$ auf Kantenklausel (a,d), ergibt sich folgende Kettenschablone$^{(4)}$:

~~$\overline{\text{Path}(\text{edge}(a,d))}$~~, $\underline{\overline{\text{Path}(\text{edge}(d,X_3))}}$, $\overline{\text{Path}(\text{edge}(X_3,X_4))}$, $\overline{\text{Path}(\text{edge}(X_4,X_5))}$, $\underline{\overline{\text{Path}(\text{edge}(X_5,a))}}$,

$COSTS_4$, $\text{Visited}(b), \text{Visited}(c),$ ~~$\text{Visited}(d)$~~, $\text{Visited}(e)$, $\text{Trace}(\text{edge}(a,d))$.

Trifft diese Kettenschablone$^{(4)}$ auf Kantenklausel (c,a), ergibt sich folgende Kettenschablone$^{(3)}$:

$\underline{\overline{\text{Path}(\text{edge}(d,X_3))}}$, $\overline{\text{Path}(\text{edge}(X_3,X_4))}$, $\underline{\overline{\text{Path}(\text{edge}(X_4,c))}}$, ~~$\overline{\text{Path}(\text{edge}(c,a))}$~~,

$COSTS_3$, $\text{Visited}(b)$, ~~$\text{Visited}(c)$~~, $\text{Visited}(e)$, $\text{Trace}(\text{edge}(a,d)), \text{Trace}(\text{edge}(c,a))$.

Trifft diese Kettenschablone$^{(3)}$ auf Kantenklausel (b,c), ergibt sich folgende Kettenschablone$^{(2)}$:

$\underline{\overline{\text{Path}(\text{edge}(d,X_3))}}$, $\underline{\overline{\text{Path}(\text{edge}(X_3,b))}}$, ~~$\overline{\text{Path}(\text{edge}(b,c))}$~~,

$COSTS_2$, ~~$\text{Visited}(b)$~~, $\text{Visited}(e)$, $\text{Trace}(\text{edge}(a,d)), \text{Trace}(\text{edge}(c,a)), \text{Trace}(\text{edge}(b,c))$.

Trifft diese Kettenschablone$^{(2)}$ auf Kantenklausel (d,e), ergibt sich folgende Kettenschablone$^{(1)}$:

~~$\overline{\text{Path}(\text{edge}(d,e))}$~~, $\underline{\overline{\text{Path}(\text{edge}(e,b))}}$,

$COSTS_1$, ~~$\text{Visited}(e)$~~, $\text{Trace}(\text{edge}(a,d)), \text{Trace}(\text{edge}(c,a)), \text{Trace}(\text{edge}(b,c)), \text{Trace}(\text{edge}(d,e))$.

Trifft diese Kettenschablone$^{(1)}$ auf Kantenklausel (e,b), ergibt sich eine Lösung:

~~$\overline{\text{Path}(\text{edge}(e,b))}$~~, $COSTS_0$,

$\text{Trace}(\text{edge}(a,d)), \text{Trace}(\text{edge}(c,a)), \text{Trace}(\text{edge}(b,c)), \text{Trace}(\text{edge}(d,e)), \text{Trace}(\text{edge}(e,b))$.

Die letzte Anwendung der separaten Multiresolution war trotz unmöglicher Resolution in Sektion *Stadtintegration* möglich, da sich die priorisierte Sektion *Lösungsfindung* durch die Resolutionsanwendung auflöst. Die hergeleitete Lösung gibt durch die *Costs*-Literale Aufschluss über ihre Kosten und durch die *Trace*-Literale Auskunft über die verwendeten elementaren Wegstücke.

Anhang B

Kostenmatrizen für TSP-Experimente

007aa - 7 Städte, asymmetrisch

	1	2	3	4	5	6	7
1	-	727	247	768	860	574	194
2	64	-	752	20	4	576	974
3	875	431	-	497	47	996	417
4	361	748	304	-	920	473	438
5	705	429	572	513	-	449	848
6	300	232	689	23	322	-	971
7	512	738	819	818	803	588	-

tspSym10 - 10 Städte, symmetrisch

	1	2	3	4	5	6	7	8	9	10
1	-	4548	1186	3648	2495	4895	1896	3530	4738	1332
2	4548	-	149	470	4190	4127	401	898	1396	1638
3	1186	149	-	1233	683	4294	395	3124	4977	1217
4	3648	470	1233	-	3446	2284	3297	836	3698	4485
5	92495	4190	683	3446	-	4668	2775	285	4192	4133
6	4895	4127	4294	2284	4668	-	4666	2296	399	2921
7	1896	401	395	3297	2775	4666	-	2786	1629	1948
8	3530	898	3124	836	285	2296	2786	-	1036	4618
9	4738	1396	4977	3698	4192	399	1629	1036	-	3281
10	1332	1638	1217	4485	4133	2921	1948	4618	3281	-

012aa - 12 Städte, asymmetrisch

	1	2	3	4	5	6	7	8	9	10	11	12
1	-	376	101	466	697	8	681	148	611	586	333	791
2	11	-	789	288	655	465	593	760	771	586	863	156
3	91	525	-	664	162	941	490	748	373	558	531	278
4	142	865	118	-	118	911	649	11	699	834	331	341
5	898	653	569	210	-	462	592	375	98	930	606	178
6	936	952	325	128	565	-	812	672	466	20	764	717
7	483	599	818	589	82	307	-	926	107	654	335	916
8	948	747	868	119	627	496	207	-	821	161	874	309
9	34	475	430	676	731	1	828	711	-	426	934	746
10	897	537	251	328	21	545	891	988	649	-	471	419
11	7	713	18	667	566	423	628	203	765	317	-	509
12	542	801	145	702	988	637	807	592	985	474	481	-

nbf15 - 15 Städte, nicht-barrierefrei

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	-	4548	1186	3648	2495	4895	1896	3530	4738	1332	-	-	-	-	-
2	4548	-	149	470	4190	4127	401	898	1396	1638	-	-	-	-	-
3	1186	149	-	1233	683	4294	395	3124	4977	1217	-	-	500	-	-
4	3648	470	1233	-	3446	2284	3297	836	3698	4485	-	-	-	-	500
5	92495	4190	683	3446	-	4668	2775	285	4192	4133	-	-	800	-	-
6	4895	4127	4294	2284	4668	-	4666	2296	399	2921	-	-	-	-	-
7	1896	401	395	3297	2775	4666	-	2786	1629	1948	-	-	-	-	-
8	3530	898	3124	836	285	2296	2786	-	1036	4618	-	-	-	-	-
9	4738	1396	4977	3698	4192	399	1629	1036	-	3281	-	-	-	-	-
10	1332	1638	1217	4485	4133	2921	1948	4618	3281	-	-	-	-	-	12
11	-	-	-	-	-	-	-	-	-	-	-	500	-	-	-
12	-	-	-	-	-	-	-	-	-	-	-	-	500	-	-
13	-	-	-	-	-	-	-	-	-	-	-	-	-	500	-
14	-	500	-	-	500	-	-	-	-	-	-	-	500	-	500
15	-	-	-	-	-	-	-	500	-	-	12	-	34	222	-

ulysses - 16 Städte, Rundreise des Odysseus

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	-	509	501	312	1019	736	656	60	1039	726	2314	479	448	479	619	150
2	509	-	126	474	1526	1226	1133	532	1449	1122	2789	958	941	978	1127	542
3	501	126	-	541	1516	1184	1084	536	1371	1045	2728	913	904	946	1115	499
4	312	474	541	-	1157	980	919	271	1333	1029	2553	751	704	720	783	455
5	1019	1526	1516	1157	-	478	583	996	858	855	1504	677	651	600	401	1033
6	736	1226	1184	980	478	-	115	740	470	379	1581	271	289	261	308	687
7	656	1133	1084	919	583	115	-	667	455	288	1661	177	216	207	343	592
8	60	532	536	271	996	740	667	-	1066	759	2320	493	454	479	598	206
9	1039	1449	1371	1333	858	470	455	1066	-	328	1387	591	650	656	776	933
10	726	1122	1045	1029	855	379	288	759	328	-	1697	333	400	427	622	610
11	2314	2789	2728	2553	1504	1581	1661	2320	1387	1697	-	1838	1868	1841	1789	2248
12	479	958	913	751	677	271	177	493	591	333	1838	-	68	105	336	417
13	448	941	904	704	651	289	216	454	650	400	1868	68	-	52	287	406
14	479	978	946	720	600	261	207	479	656	427	1841	105	52	-	237	449
15	619	1127	1115	783	401	308	343	598	776	622	1789	336	287	237	-	636
16	150	542	499	455	1033	687	592	206	933	610	2248	417	406	449	636	-

ft53 - 53 Städte, asymmetrisch

Teil 1:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-	223	210	125	245	363	322	388	290	341	267	281	258	247	302	325	311	276
2	58	-	179	100	287	155	116	170	74	137	239	260	234	221	282	294	286	251
3	60	109	-	80	295	246	209	270	174	225	227	179	221	212	262	131	117	79
4	58	115	98	-	285	255	215	279	178	232	154	176	157	136	194	219	207	170
5	93	58	63	90	-	194	160	210	117	174	238	235	231	224	280	175	171	133
6	531	593	584	499	765	-	344	405	309	360	631	654	632	622	679	697	685	647
7	417	484	469	384	651	55	-	257	190	180	519	547	524	506	563	586	573	533
8	466	521	506	427	693	86	111	-	236	275	566	585	505	546	552	621	618	574
9	243	309	285	205	468	92	54	111	-	69	342	366	345	326	385	406	391	350
10	262	321	302	222	491	81	68	99	32	-	359	386	357	340	400	424	409	371
11	361	416	404	317	435	522	512	544	471	449	-	194	179	327	228	441	438	392
12	182	241	229	134	411	376	341	402	302	314	114	-	195	184	246	304	288	258
13	271	305	315	224	271	354	341	374	301	283	110	105	-	164	60	276	263	233
14	216	277	259	176	281	209	199	227	155	140	35	54	24	-	77	132	125	79
15	275	338	327	242	344	300	293	331	256	241	122	119	99	111	-	230	220	181
16	369	329	451	411	404	405	367	422	324	378	345	393	514	539	561	-	329	297
17	246	305	286	204	416	353	317	374	276	331	181	75	266	252	307	24	-	310
18	278	343	325	243	450	390	353	417	315	367	215	115	301	293	349	65	48	-
19	118	85	249	161	356	148	113	176	78	128	250	149	299	284	345	70	86	44
20	134	93	266	176	364	172	137	191	90	153	255	155	317	301	363	93	81	62
21	306	367	350	263	371	212	171	229	139	186	238	142	322	312	365	412	417	378
22	376	460	449	354	466	305	266	323	227	282	336	230	416	408	462	448	448	424
23	136	225	336	245	369	292	264	314	221	277	328	231	387	368	428	221	210	191
24	223	307	374	280	399	233	198	252	158	213	256	160	343	332	388	303	298	273
25	183	267	381	290	407	296	267	319	223	276	329	226	411	402	460	256	251	230
26	598	658	641	559	823	621	644	551	718	780	695	718	695	682	739	689	678	707
27	536	605	586	502	765	567	585	494	669	719	635	657	639	627	680	630	622	652
28	581	646	628	543	812	611	638	542	710	767	687	701	682	673	724	675	664	698
29	620	677	670	581	853	647	655	579	624	677	717	620	716	702	761	718	701	734
30	586	647	632	545	821	614	634	547	663	716	692	668	681	674	735	681	672	698
31	500	712	702	613	737	847	812	871	776	829	756	770	747	737	792	812	803	764
32	619	720	707	621	850	685	705	614	723	772	764	725	756	746	807	758	739	766
33	768	870	853	766	1000	831	853	762	872	923	905	870	910	892	956	904	893	916
34	722	816	801	722	951	786	805	710	820	873	860	820	859	848	901	858	847	875
35	730	693	861	768	883	538	493	738	678	664	619	759	779	894	825	678	689	652
36	341	300	311	337	257	375	338	399	305	361	254	374	427	467	474	297	306	267
37	371	325	341	374	292	403	360	425	331	382	139	312	308	451	352	323	335	303
38	247	200	330	286	276	274	236	300	203	257	218	273	383	410	434	192	207	168
39	225	179	195	223	140	320	280	337	249	301	147	321	321	346	367	250	264	221
40	331	288	305	330	253	178	139	379	292	304	221	360	395	459	433	289	301	261
41	250	214	223	255	174	287	245	305	209	262	165	283	335	373	385	211	218	177
42	529	587	576	488	738	428	456	355	574	613	548	618	623	605	669	593	602	569
43	412	441	461	373	592	269	287	194	421	448	495	512	509	497	556	437	447	406
44	393	453	440	353	622	347	370	272	497	527	498	519	495	478	544	516	527	488
45	464	520	508	421	648	353	370	275	401	464	505	407	558	546	605	520	533	489
46	296	360	344	258	527	348	375	276	427	477	394	415	398	385	440	462	449	406
47	339	403	387	301	524	359	387	286	443	495	435	462	436	426	479	443	453	409
48	364	426	411	320	564	276	299	211	425	461	379	445	465	453	511	448	460	423
49	1551	1527	1596	1509	1784	1572	1562	1504	1522	1513	1653	1593	1650	1634	1695	1512	1530	1489
50	1599	1569	1649	1566	1834	1527	1515	1543	1478	1463	1706	1641	1621	1687	1673	1571	1578	1543
51	1504	1464	1629	1541	1713	1598	1564	1542	1528	1543	1552	1549	1454	1603	1497	1550	1557	1526
52	1505	1534	1545	1464	1705	1550	1572	1487	1533	1523	1607	1601	1548	1586	1590	1532	1543	1503
53	1520	1472	1612	1526	1722	1589	1574	1517	1528	1520	1559	1558	1466	1616	1513	1530	1536	1501

ft53 - 53 Städte, asymmetrisch

Teil 2:

	19	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
1	392	475	417	324	372	437	610	310	362	333	498	459	467	352	199	336	305	172
2	434	512	409	367	406	410	598	354	353	371	451	490	510	389	244	313	291	210
3	382	431	263	309	321	235	452	355	338	324	497	449	360	243	251	140	272	223
4	429	516	326	366	411	323	517	344	270	368	541	498	446	323	244	225	203	214
5	281	267	221	248	139	286	415	82	136	135	309	264	341	293	231	189	75	253
6	909	992	807	847	887	809	999	789	750	848	763	716	763	643	724	539	682	687
7	803	875	693	733	777	692	888	607	577	662	585	539	575	454	611	361	561	581
8	804	863	594	495	733	735	790	695	673	660	675	627	674	556	654	453	615	614
9	619	705	510	552	599	511	707	501	451	558	472	425	471	346	432	253	394	393
10	639	714	536	569	611	528	728	448	415	504	424	376	412	296	450	196	407	419
11	480	560	357	401	554	557	549	502	453	418	596	544	655	532	383	455	396	352
12	499	583	376	420	494	411	568	395	315	433	612	564	511	413	364	311	250	333
13	313	400	186	241	395	389	387	336	283	255	424	378	486	361	220	290	224	179
14	331	411	204	253	329	246	400	225	147	265	443	397	346	239	229	140	80	193
15	400	479	144	197	371	340	337	327	242	204	384	334	446	347	295	238	176	265
16	264	313	405	312	528	446	596	407	358	316	380	448	557	432	286	353	460	163
17	275	324	158	207	225	136	356	337	270	221	398	348	471	348	204	368	315	177
18	308	365	195	243	261	175	388	381	292	260	430	392	292	170	237	79	358	217
19	-	90	226	277	293	204	425	180	241	241	162	363	330	210	271	109	355	227
20	31	-	227	276	291	207	418	102	160	162	82	291	341	227	279	118	370	242
21	356	371	-	64	239	275	207	196	121	84	252	204	558	432	285	370	375	226
22	389	372	115	-	250	262	307	291	225	171	350	306	600	483	332	468	472	322
23	160	145	101	122	-	251	300	235	212	168	211	299	472	353	318	248	239	296
24	247	226	36	87	96	-	233	217	147	100	270	227	350	236	84	331	328	242
25	200	185	103	92	61	81	-	270	219	173	254	297	420	302	153	288	278	316
26	946	998	734	782	890	806	924	-	71	72	244	195	729	609	789	715	750	756
27	887	942	766	819	831	748	960	105	-	158	334	284	811	700	726	709	690	692
28	925	982	675	727	871	789	866	135	61	-	184	141	669	547	771	651	722	737
29	835	853	498	547	725	763	696	36	99	98	-	218	500	375	773	479	775	618
30	880	896	547	598	773	791	741	85	67	138	63	-	546	429	780	522	736	662
31	879	958	908	814	861	925	1103	797	782	822	987	943	-	653	691	559	793	659
32	883	862	596	502	740	747	791	220	134	281	447	407	136	-	485	118	815	775
33	1032	1015	742	648	884	901	936	369	279	431	598	551	285	166	-	262	964	931
34	978	959	702	600	837	852	894	322	232	376	554	510	235	119	381	-	915	879
35	619	706	838	784	898	818	1026	520	435	582	752	705	274	316	169	394	-	642
36	237	315	264	163	384	416	462	291	215	178	349	302	461	345	194	332	324	-
37	266	345	297	195	413	442	485	315	246	204	373	333	341	222	81	329	351	48
38	139	218	282	179	395	324	472	303	231	185	280	321	428	309	157	230	334	40
39	188	268	301	214	265	382	499	203	265	214	337	345	361	241	88	277	202	61
40	227	311	304	206	374	415	491	311	255	215	370	335	433	315	166	317	315	96
41	152	226	170	79	297	331	365	204	132	85	260	207	373	256	109	243	234	54
42	537	613	494	548	561	477	693	403	454	461	380	590	556	594	445	628	289	509
43	373	454	427	480	490	410	628	339	400	398	322	530	496	532	381	464	229	360
44	452	537	459	515	520	442	659	426	478	475	399	606	528	560	413	547	261	433
45	457	478	282	334	346	262	483	425	398	353	401	479	343	387	234	464	86	447
46	459	534	360	410	430	338	552	422	475	426	398	551	426	463	315	465	162	442
47	377	457	354	412	422	331	554	387	443	420	364	546	423	456	310	473	160	283
48	386	469	314	364	378	295	510	220	278	282	203	410	383	414	272	478	113	327
49	1455	1532	1512	1560	1549	1582	1505	1050	1026	1104	1028	978	1419	1297	1562	1197	1678	1635
50	1504	1585	1548	1530	1496	1520	1458	1102	1078	1154	1079	1027	1360	1248	1506	1144	1628	1682
51	1490	1566	1539	1566	1579	1609	1534	1080	1062	1144	1060	1010	1192	1071	1343	977	1621	1627
52	1470	1547	1519	1573	1563	1549	1517	1057	1045	1120	1040	993	1280	1168	1432	1060	1366	1462
53	1466	1544	1517	1573	1562	1593	1518	1056	1044	1116	1036	992	1206	1088	1352	986	1469	1571

ft53 - 53 Städte, asymmetrisch

Teil 3:

	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
1	143	273	292	364	262	296	435	423	433	354	509	472	238	185	357	420	344
2	185	318	329	402	297	81	241	215	292	308	301	262	207	159	300	214	314
3	189	259	337	388	308	174	340	309	382	396	393	355	198	141	201	302	190
4	183	314	278	403	301	183	352	311	391	341	410	367	130	76	247	312	236
5	219	306	360	351	340	123	277	248	288	323	305	256	155	155	152	248	146
6	664	788	762	881	781	669	832	796	824	825	885	838	90	140	164	70	296
7	543	682	645	769	666	546	577	565	493	662	653	700	123	177	201	112	343
8	587	685	634	715	707	591	673	659	589	690	749	771	154	209	228	147	371
9	373	500	468	588	491	369	532	501	535	534	591	549	160	210	233	147	376
10	386	521	488	605	503	389	554	521	554	550	614	568	151	204	228	137	359
11	319	359	313	475	437	352	453	467	453	373	527	531	78	135	308	392	291
12	305	375	322	493	425	306	469	435	466	385	531	481	180	202	333	425	319
13	154	197	138	308	273	182	290	308	285	203	363	362	179	231	311	399	293
14	163	211	157	322	288	200	296	320	295	215	374	380	94	148	163	255	151
15	235	275	223	262	352	140	303	270	346	286	363	315	189	242	261	352	252
16	226	141	278	276	244	379	425	443	412	335	494	538	86	146	89	188	75
17	238	151	288	279	260	143	307	267	356	351	362	316	99	160	103	199	94
18	274	193	327	318	294	184	346	312	394	384	405	355	138	192	141	228	127
19	245	198	337	344	306	146	307	275	359	379	368	327	146	202	154	216	140
20	261	216	351	351	322	169	325	294	374	392	390	338	157	203	151	225	142
21	223	227	248	138	265	351	377	413	389	304	409	368	285	285	330	268	310
22	322	327	342	231	361	429	383	426	401	407	415	372	356	306	421	370	410
23	266	314	333	230	351	199	149	187	173	203	182	141	121	72	239	339	236
24	246	253	264	165	288	280	238	269	255	291	268	217	207	156	332	293	320
25	303	319	339	226	352	241	195	232	208	245	226	181	171	117	289	363	272
26	721	819	819	857	838	727	689	726	716	744	718	675	315	368	321	407	309
27	660	762	768	882	780	666	779	803	799	827	813	768	399	460	404	499	396
28	713	809	808	801	833	682	637	677	650	688	665	620	257	312	259	348	246
29	711	711	734	621	714	751	713	752	733	768	743	701	86	138	91	182	80
30	714	763	778	669	757	715	762	800	781	815	796	752	132	184	135	224	119
31	628	764	774	849	753	784	927	911	917	838	998	963	724	677	843	911	833
32	752	815	830	717	853	673	700	620	795	721	778	815	520	573	528	621	514
33	898	966	983	866	1000	819	854	775	946	870	926	964	669	725	676	766	664
34	852	912	931	825	950	773	797	722	902	821	871	923	625	678	621	712	611
35	698	617	755	754	722	756	896	889	889	815	974	932	95	144	99	188	84
36	135	116	137	210	109	238	275	296	272	199	358	396	124	73	243	335	231
37	-	142	158	231	133	260	307	322	301	226	386	423	152	102	271	366	264
38	93	-	154	158	120	254	293	311	291	216	367	408	148	98	269	337	261
39	23	67	-	207	147	117	157	174	152	76	238	271	168	116	279	371	268
40	98	110	125	-	144	225	267	285	265	188	341	383	215	162	329	241	317
41	47	26	48	118	-	143	188	205	181	108	260	306	167	116	279	341	271
42	566	476	613	616	586	-	172	144	227	241	232	189	257	204	370	466	358
43	418	335	464	470	436	61	-	107	158	135	84	131	212	168	309	323	297
44	497	410	551	553	515	67	94	-	190	115	163	211	128	74	245	338	239
45	497	415	518	408	528	65	98	76	-	181	170	221	161	147	167	260	152
46	423	422	526	491	523	50	94	112	94	-	173	213	226	178	240	330	232
47	338	264	394	397	369	31	110	29	88	58	-	184	144	97	240	329	221
48	383	306	434	443	406	76	26	67	47	86	61	-	177	134	193	294	184
49	1678	1643	1747	1637	1717	1578	1617	1634	1614	1534	1693	1672	-	73	233	330	224
50	1727	1689	1754	1666	1775	1517	1568	1582	1557	1486	1638	1621	60	-	188	274	169
51	1593	1632	1577	1663	1715	1514	1553	1578	1548	1477	1627	1670	44	100	-	272	159
52	1581	1563	1584	1648	1551	1255	1302	1319	1298	1224	1380	1412	24	79	106	-	240
53	1607	1642	1595	1643	1658	1362	1411	1429	1401	1325	1479	1516	21	80	29	118	-

Index