Selbstorganisation in einem System von Binärstrings mit algorithmischen Sekundärstrukturen

- Diplomarbeit -

im

Fachbereich Informatik
der Universität Dortmund

vorgelegt von

Peter Dittrich[1]

29. Mai 1995

Erstgutachter und Betreuer	:	Prof. Dr. W. Banzhaf
Zweitgutachter	:	Prof. Dr.-Ing. H.-P. Schwefel

Universität Dortmund
Fachbereich Informatik
Lehrstuhl für Systemanalyse
D-44221 Dortmund

[1] e-mail: dittrich@LS11.informatik.uni-dortmund.de

Inhaltsverzeichnis

Einleitung

Die Frage nach der Entstehung des Lebens und der Entwicklung der beeindruckenden Vielfalt und Sinnhaftigkeit seiner Natur ist weiterhin unbeantwortet. Die millionenfache Dimensionalität des Variantenraums offenbart, daß eine blinde Suche, getrieben vom Darwinschen Prinzip der Variation und Selektion, als Erklärung völlig unzureichend ist. Aktuelle Erklärungsversuche inkorporieren Konzepte der Selbstorganisation und der spontanen Strukturbildung. Die Validierung solcher Theorien ist problematisch, da keine brauchbaren Fragmente der präbiotischen Evolution die Jahrmillionen überdauert haben. Auch Rekonstruktionsversuche im Labor haben bisher keine befriedigenden Ergebnisse geliefert.

Abhilfe bieten Computerexperimente. Allerdings wird nicht eine detaillierte Simulation der physikalischen Vorgänge angestrebt, sondern die Instanziierung einer künstlichen Welt in einem Computersystem, wobei die dort vorhandenen "natürlichen" Fähigkeiten respektiert und effizient genutzt werden. Dieses Vorgehen macht nur Sinn, falls die grundlegende Arbeitshypothese gilt, daß die für lebende Systeme typischen Phänomene unabhängig von ihrer materiellen Basis sind.

Die materielle Basis in dieser Arbeit sind endliche Sequenzen von Nullen und Einsen – sogenannte **Binärstrings**. Die Welt, in der sie agieren, ist ein gut gerührter algorithmischer Reaktor, der hier mit 100 bis 10^6 Binärstrings gefüllt sein kann. Zwei Strings können miteinander kollidieren und dabei einen neuen bilden. Dazu wird ein String in seine **Sekundärstruktur** überführt, die einer Maschine entspricht, welche den anderen String als Eingabe verarbeitet und das Reaktionsprodukt als Ausgabe liefert. Zur Beschreibung der Sekundärstruktur wird ein endlicher formaler Automat verwendet, dessen Funktionalität eindeutig durch die Primärstruktur des Strings determiniert ist.

Nachdem in Kapitel 1 der Begriff der Selbstorganisation diskutiert worden ist und eine Einordung der Arbeit in bestehende Arbeitsgebiete erfolgt, wird in Kapitel 2 das Konzept des algorithmischen Reaktors präzisiert und für einen Spezialfall ein mathematisches Modell vorgestellt.

Das Kapitel 3 stellt alternative algorithmische Sekundärstrukturen vor und definiert am Ende die "Automaten-Reaktion", die für die Simulationen im Kapitel 4 verwendet wird. Kapitel 4 ist der Kern der Arbeit. Es präsentiert ausführlich die beobachteten Phänomene der Selbstorganisation, Evolution und Veränderung der Evolutionsfähigkeit, die sich unter anderem in der spontanen Ausbildung eines Crossover-Mechanismus gezeigt haben. Dies ist insofern ungewöhnlich, da keine expliziten Mutations-, Rekombinations- oder künstliche Selektionsoperatoren verwendet werden.

In Kapitel 5 wird anhand von Beispielen gezeigt, daß sowohl algorithmische als auch chemische Reaktoren in der Lage sind, Informationen zu speichern und zu verarbeiten.

Kapitel 1

Grundlagen

1.1 Organisation und Selbstorganisation

Organisation

Der Begriff "Organisation" wird sowohl in der Umgangssprache als auch in den Fachsprachen der Wissenschaften vielfältig verwendet. Eine Flut von Definitionen, Präzisierungen und Eingrenzungen des Organisationsbegriffs in speziellen Arbeitsgebieten und Forschungsrichtungen ist also kaum verwunderlich. Zum Beispiel findet man in der Betriebswirtschaftslehre einen instrumentalen Organisationsbegriff. Dabei wird das System der Verhaltensregeln von Personen und Funktionsregeln von Maschinen einer Unternehmung als **Organisation** bezeichnet ([Grochia 1978] S. 12, [Welge 1987] S. 2). Das Aufstellen dieser Regelmenge nennt man **organisieren**.

Im folgenden soll der Terminus "Organisation" allgemein für beliebige Systeme verwendet werden. Ein **System** besteht aus einer Menge von **Komponenten** und **Relationen**, die einen Zusammenhang zwischen *allen* Komponenten herstellen. Die Beschreibung eines Systems durch Aufzählung seiner Komponenten und Relationen kann umständlich werden, falls diese sich mit der Zeit ändern. Es werden deshalb oft **Systemgrenzen** dem System zugeschlagen. Zum Beispiel ändert sich die Zusammensetzung des Systems "Personen, die die Regierung bilden" laufend, und in diesem speziellen Fall ist sogar für keine Komponente die Systemzugehörigkeit auf Dauer gesichert.

Jetschke nennt **Organisation** "das funktionale Zusammenspiel der Elemente zu einer das ganze System betreffenden Wirkung" ([Jetschke 1989] S. 15). In der Unternehmung ist zum Beispiel das "funktionale Zusammenspiel" durch die Struktur der Weisungsbefugnisse bestimmt (vgl. [Welge 1987] S. 481 ff.).

Es ist auch möglich, "Organisation" formal zu definieren. Derartige Versuche basieren auf formalen Systemtheorien. Zum Beispiel betrachtet Schwegler ein System als ein Netz von **Relatoren**. Relatoren sind nicht Komponenten des Systems, sondern "Elemente der Beschreibung", was einen Beobachter voraussetzt. Die Einbeziehung des Beobachters macht Sinn, denn Systeme hängen von der Betrachtungsweise eines Beobachters ab, wo er Systemgrenzen zieht und welche Komponenten und Relationen er dem System als

zugehörig erachtet. **Organisation** ist dann die "Gesamtheit der Gesetze, die bei einer Zusammensetzung aus Relatoren eine Rolle spielen" ([SCHWEGLER 1992] S.42).

Präzisierungen wie die von SCHWEGLER sind wegen des erforderlichen aufwendigen theoretischen Unterbaus unhandlich und bringen für die vorliegende Arbeit keinen Vorteil. Deshalb genügt es völlig, **Organisation** als die relativen Beziehungen und Abhängigkeiten der Komponenten untereinander zu betrachten, deren Wirkung in Ordnungsphänomenen sichtbar wird. Unter **Ordnung** sollen hervorstechende beobachtbare Regelmäßigkeiten innerhalb vorhandener Unterschiede verstanden werden ([JETSCHKE 1989] S.15), deren Bestimmung aber kontextabhängig ist. Ordnung ist ein erkenntnissubjektiver Begriff und abhängig von einem Beobachter (s. ausführlich die Sammlung [KROHN UND KÜPPERS 1992] und darin besonders [MÜLLER-HEROLD 1992]). Ordnung könnte man als qualitative Wirkung der Organisation auffassen.

Selbstorganisation

Auch für den Terminus **Selbstorganisation** genügt hier eine allgemeine Anschauung, wie sie zum Beispiel von AN DER HEIDEN vertreten wird. Danach sind alle Eigenschaften und Phänomene eines Systems **selbstorganisiert**, die durch die dynamischen Abhängigkeiten seiner Komponenten hervorgebracht und nicht von außen aufgeprägt oder aufgezwungen werden. Diese Vorstellung geht also davon aus, daß Selbstorganisation durch Entitäten des Systems selbst bewirkt wird und nicht von einem *deus in machina*. **Selbstorganisation** bezieht sich auf Phänomene, die in Systemen und deren Strukturen infolge der Interaktion ihrer Komponenten auftreten ([AN DER HEIDEN 1992] S. 72). Der Vorteil einer solchen Auffassung von Selbstorganisation ist, daß sie 1.) sich mit der umgangssprachlichen Verwendung deckt, 2.) allgemein gehalten ist, um nicht von vornherein auf spezielle naturwissenschaftliche Gebiete beschränkt zu sein und 3.) sich auf der Basis der Theorie dynamischer Systeme mathematisch formalisieren läßt, wie in [AN DER HEIDEN 1992] gezeigt wird.

Bei starker interner Dynamik und erheblich schwankenden externen Einflüssen kann eine Unterscheidung zwischen Selbstorganisation und Fremdorganisation unmöglich werden. Deshalb werden im Rahmen dieser Arbeit möglichst alle externen Einflüsse vermieden, so daß die beobachteten Phänomene der Selbstorganisation zugeschrieben werden können.

Problematisch ist auch der relative Charakter der Selbstorganisation. Betrachtet man zum Beispiel die vorliegende Arbeit als System, so ist der Inhalt nicht selbstorganisiert, da er nicht durch die Interaktion der Systemkomponenten (Seiten, Umschlag, Moleküle etc.) zustande gekommen ist. Werden aber die Systemgrenzen erweitert, um den Autor und die entsprechenden Werkzeuge (Rechner, Fotokopierer, Bibliothek etc.) hinzuzunehmen, so ist der Prozeß der Entstehung dieser Zeilen selbstorganisiert. Die Situation ist allerdings nicht so eindeutig, falls man miteinbezieht, daß Dinge, wie zum Beispiel die Formen der Schriftzeichen, nicht eine Erfindung des Autors sind und der Inhalt durch externe Einflüsse beeinflußt worden ist.

Für die hier verwendete Vorstellung von Selbstorganisation gilt also: 1.) Ob eine Eigenschaft selbstorganisiert ist, hängt von den Systemgrenzen ab. 2.) In der Realität gibt es praktisch keine *reinen* selbstorganisierten oder fremdorganisierten Systeme.

Selbstorganisation und dissipative Strukturen

Nicht unerwähnt bleiben soll eine speziellere Verwendung des Begriffs der Selbstorganisation, die sich aus Arbeiten von ANDRONOW, TURING, PRIGOGINE, HAKEN und anderen ergibt (z.B. [HAKEN 1977]). Der Terminus "Selbstorganisation" wird dort im Zusammenhang mit **dissipativen Systemen** (genauer: dissipativen Strukturen) verwendet. Dabei handelt es sich um offene oder geschlossene Systeme, die durch einen ständigen Stoff- oder Energieaustausch, fern vom thermodynamischen Gleichgewicht, geordnete Strukturen aufbauen können. Die dissipative Struktur kann nur existieren, solange das System mit hochwertiger Energie (z.B. kinetischer, chemischer oder elektrischer Energie) versorgt wird und niederwertige Energie in Form von Wärme abführen kann. Das System setzt also Energie um, um einen gewissen Ordnungsgrad aus sich selbst heraus aufzubauen und zu erhalten. Klassische Beispiele hierfür sind die Konvektionszellen beim Bénard-Effekt und die Emmission kohärenten Lichts beim Laser [HAKEN 1970]. Gegenbeispiele sind, ein frei schwingendes Pendel und gefrierendes Wasser. Beide Gegenbeispiele sind allerdings unter obiger allgemeinen Auffassung Phänomene der Selbstorganisation.

Eiskristalle sind ein Beispiel für **konservative Strukturen**, die im thermodynamischen Gleichgewicht durch Abkühlen und die damit verbundene verstärkte Wirkung konservativer Kräfte entstehen.

Ein Problem der Theorie dissipativer Systeme besteht darin, daß noch keine befriedigende Übertragung auf algorithmische informationsverarbeitende Systeme existiert.

1.2 Motivation und Philosophie

Eine **Philosophie** kann von äußerst praktischer Natur sein, falls man darunter die Summe der grundlegenden Wertvorstellungen versteht (wie z.B. [WELGE 1985] S. 19-25). Die grundlegenden Werte werden in einem **Philosophiespektrum** zusammengefaßt, das sich dann praktisch verwenden läßt, 1.) zur Ableitung eines Zielsystems, 2.) zur Entscheidungsunterstützung bei Designvarianten, die zur Auswahl stehen, 3.) als Hilfe für den Leser, um Motivation und Konstruktionsentscheidungen nachvollziehen zu können und 4.) als Koordinierungsinstrument, falls mehrere Personen involviert wären.

Es folgt eine Übersicht über das **Philosophiespektrum**, das dieser Arbeit zugrunde liegt.

1. **Einfachheits-Prinzip** (engl. keep-it-simple, KIS)
   Das Einfache ist dem Komplexen vorzuziehen. Bei der Durchführung der Experimente ist darauf zu achten, daß das zugrundeliegende System möglichst einfach ist, um so einen Zusammenhang zwischen beobachteter Wirkung und Ursache herstellen zu können.

2. **Respektiere das Medium** (engl. respect the medium)
   Designentscheidungen sollen durch reale physikalische Systeme (biologischer und besonders technischer Art) motiviert sein, die für die Implementierung in Frage kommen. Mit **Medium** ist hier ein technisches informationsverarbeitendes System gemeint. Es sollen also die Vorzüge und Beschränkungen eines verfügbaren Rechners respektiert werden. Zum Beispiel ist die Handhabung natürlicher Zahlen wesentlich

einfacher als die reeller Zahlen in Gleitpunktdarstellung. Auch ist der globale Zugriff von Information ein bekannter Flaschenhals (nicht nur bei Rechnern mit von Neumann– oder Harvard-Architektur), der vermieden werden soll. Konkret kann man sich unter dem Medium die Architekturebene oder auch die Gatterebene vorstellen.

3. **Lokalitätsprinzip**
   Es gibt keine oder eine möglichst geringe globale Kontrolle. Das Verhalten des gesamten Systems soll durch lokale Wechselwirkung vieler relativ einfacher Komponenten entstehen. Will man den von Neumannschen-Flaschenhals (engl. bottleneck) umgehen, so ist das nur mit hochparallelen Rechnerstrukturen möglich. Darunter fallen die Connection Machine [Hillis 1985], neuronale Rechner (z.B. die Synapse), zelluläre Rechner (z.B. MITs CAM8), molekulare Computer [Banzhaf 1990; Adleman 1994] und Quantum Computer [Deutsch 1985; Maddox 1985; Feynman 1986]. Um eine adäquate Skalierbarkeit, Robustheit und Fehlersicherheit zu gewährleisten muß auf globale Kontrollen verzichtet werden [Nussbaum und Agarwal 1991; Sun und Rover 1994]. Die Hauptarbeit wird durch lokale asynchrone Operationen erledigt.

Das Lokalitätsprinzip liegt auch den Gebieten Artificial Life (kurz: AL), Adaptive Behavior und der Animate-Bewegung zugrunde [Meyer und Guillot 1994]. Das Einfachheits-Prinzip wird aber dort nicht generell berücksichtigt (z.B. [Cliff 1991]). Die Forderung, das Medium "Computer" so zu respektieren, wie es ist, stammt von Ray. Sein TIERRA-System besteht im wesentlichen aus einem virtuellen Computer mit einem linearen Speicher, in dem Programme "leben", die sich reproduzieren und mit der Zeit weiterentwickeln [Ray 1991].

1.2.1 Artificial Life: Geschichte

Schon immer hat das Lebendige den Menschen fasziniert. Er versuchte, es zunächst in starren Bildern und Skulpturen festzuhalten. Später (18. Jahrhundert) wurde den starren Imitationen mit Hilfe hochkomplexer Mechanik ein "Eigenleben" gegeben. Jacques de Vaucanson's mechanische Ente (1735) konnte essen, trinken, verdauen, quaken und mit den Flügeln schlagen. Menschlich wirkende Schreibautomaten, ausgestattet mit einer variablen mechanischen Steuerung, konnten zeichnen und schreiben (vgl. [Langton 1989a] S. 9 f.).

Anfang des 20. Jahrhunderts erfolgte eine Abstraktion der mechanischen "Rechenvorgänge" mit Hilfe der Mathematik und Logik (Church, Kleen, Gödel, Turing, Post etc.). Das dynamische Verhalten einer Maschine konnte unabhängig von der realen Substanz formuliert werden. Auch das Leben wurde formalisiert sowohl auf hoher Ebene von Populationen und Gemeinschaften als auch auf molekularem Level.

Von Neumann sah in der Replikationsfähigkeit eine wichtige Eigenschaft der Lebewesen. Er versuchte, das Phänomen der Selbstreplikation unabhängig von der materiellen Basis zu verstehen. In seinem berühmten Werk "Theory of Self-Reproducing Automata", das erst nach seinem Tode veröffentlicht worden ist, zeigt er in einem formalen zellulären Automaten nicht-triviale selbst-replizierende Strukturen [von Neuman 1966]. Allerdings muß erwähnt werden, daß die grundsätzliche Idee der Selbstbezüglichkeit schon bei Gödel

in dem Beweis seines bekannten Unvollständigkeitssatzes auftaucht [GÖDEL 1931; HOFSTADTER 1985].

In den darauffolgenden Jahren gab es eine Vielzahl von Arbeiten, die man in das Gebiet AL einordnen könnte: LINDENMAYERs formale Systeme zur Beschreibung zellulärer Interaktion und Entwicklung, PENROSEs selbstreplizierende Mechaniken, CONWAYs LIFE und DEWDNEYs Krieg der Kerne (engl. Core Wars), um nur einige zu nennen.

Der Mißstand der geringen Kommunikation unter den Wissenschaftlern und der mangelhaften Öffentlichkeit der Forschung wurde mit dem ersten AL Workshop [LANGTON 1989B] aus dem Wege geräumt.

1.2.2 Artificial Life

"**Artificial Life** beschäftigt sich mit künstlichen (*man-made*) Systemen, die ein Verhalten zeigen, das für natürliche lebende Systeme charakteristisch ist." ([LANGTON 1989A] Seite 1). Das natürliche Leben auf der Erde wird als eine Ausprägung einer höheren, allgemeineren Phänomenklasse angesehen, wobei die Existenz weiterer Formen von Leben denkbar ist. Die Biologie – als die Wissenschaft natürlicher lebender Systeme – muß bei der Charakterisierung dieser Phänomenklasse versagen, da ihr nur ein Beispiel vorliegt – nämlich das natürliche Leben auf unserer Erde. Durch Simulation oder sogar Realisierung von künstlichem Leben könnte eine geeignete Grundlage für das Verstehen der allgemeinen Klasse lebender Systeme geschaffen werden.

Die logische Form einer Maschine (Programm/Algorithmus) läßt sich von der physikalischen Hardware trennen. Analog geht man davon aus, daß die logische Form des Lebens unabhängig von der physikalischen Basis ist. Leben wird nicht als Eigenschaft einer Ansammlung von Materie angesehen, die auf eine bestimmte Art und Weise zusammengesetzt ist, sondern als Eigenschaft der *Organisation* der Materie ([LANGTON 1989A] Seite 2). Das heißt, Leben wird durch Verhalten charakterisiert, *nicht* durch die materielle Basis.

Die verwendete Methode ist die **Synthese** und nicht die Analyse. Die klassische Biologie sieht einen Organismus als ein komplexes Gebilde an, dessen materielle Basis es in einer *Top-Down-Analyse* zu verstehen und zu klassifizieren gilt. Artificial Life erforscht die formale Basis von Leben durch eine *Bottom-Up-Synthese*.

Das Schlüsselkonzept lautet **emergentes Verhalten** (engl. **emergent behavior**), wobei dieses Verhalten durch lokale Wechselwirkungen und nicht durch globale Regeln bestimmt ist. Konkret besteht ein AL-System aus einer großen Anzahl von einfachen Systemen, deren Zusammenwirken ein äußerst komplexes Verhalten hervorruft, das nicht explizit modelliert wurde und im allgemeinen auch nicht vorhersagbar ist. Klassische Beispiele sind zelluläre Automaten, Lindenmayer-Systeme und boolesche Netzwerke.

Wie erklärt man das Auftreten des Verhaltens lebender Systeme? Dazu wird das Genotyp-Phänotyp-Konzept der Biologie erweitert. Unter dem **verallgemeinerten Genotyp** versteht man eine große ungeordnete Menge von relativ einfachen Regeln (engl. low-level rules) eventuell zuzüglich eines Anfangszustandes. Der **verallgemeinerte Phänotyp** ist das Verhalten bzw. die Struktur, die aus den Interaktionen der einfachen Regeln hervorgeht, wenn sie in einer bestimmten Umgebung (engl. environment) aktiviert werden ([LANGTON 1989B] Seite 22). Die Entwicklung des Phänotyps aus dem Genotyp hat folgende Eigenschaften:

1. Unvorhersagbarkeit:
   Das Verhalten ist nicht auf einfache Weise aus den Regeln ableitbar. Man muß das System "laufen lassen" und beobachten.

2. Unumkehrbarkeit:
   Ein Phänotyp läßt sich nicht im allgemeinen auf den Genotyp rücktransformieren. Eine solche Transformation entspräche in der Biologie der (genetischen) Lamarckschen Evolution. Man ist also nicht allgemein in der Lage, für ein gewünschtes gegebenes Verhalten die entsprechenden Regeln anzugeben.

Um "interessante" Genotypen zu finden, wurden bisher hauptsächlich Probierverfahren angewandt. Unterstützung erhofft man sich von evolutiven Verfahren, die ihre Kraft aus dem Darwinistischen Prinzip der natürlichen Selektion und dem Prinzip der Selbstorganisation [Kauffman 1990; Kauffman 1993] schöpfen.

1.2.3 Wozu soll Artificial Life gut sein ?

Eine oft gestellte Frage, die wie folgt beantwortet werden kann:

1. Natürliches biologisches Leben verstehen:
   Das versucht die Biologie selbstverständlich ebenfalls. AL bietet aber über die Synthese alternativer Lebensformen neue Perspektiven.

2. Verstehen von künstlichem Leben:
   Vorgänge in AL-Systemen, die ein komplexes Eigenverhalten zeigen, sind es wert, wegen ihrer bloßen Existenz untersucht zu werden.

3. Globale Anwendung:
   Eine Familie, ein Land, eine Kultur oder die Erde können als ein Organismus – oder Meta-Organismus – angesehen werden [Miller 1978]. Somit könnte AL zur Erklärung gesellschaftlicher Zusammenhänge oder Vorhersage von Katastrophen in komplexen ökologischen und ökonomischen Systemen eingesetzt werden.

4. Praktische technische Anwendungen:
   Man hofft, Erkenntnisse zur Lösung technischer Problemstellungen einsetzen zu können. Vielversprechende Anwendungsgebiete sind: autonome Systeme bzw. Roboter (Brooks, Steels, Mataric, etc. in [Cliff, Husbands, Meyer, und Wilson 1994]), Unterhaltung (Sim City, Sim Life, Core Wars etc.), Computergrafik bzw. Kunst (z.B. [Todd und Latham 1991], [Sims 1994], [Terzopoulos, Tu, und Grzeszczuk 1994]) und – was nicht weiter überrascht, aber bedenklich stimmen sollte – Waffen- und Kriegstechnik (z.B. natürlich wirkende Gegner in Panzersimulatoren). Das Anwendungspotential im Unterhaltungs– bzw. Grafiksektor darf in keinem Fall unterschätzt werden. Denn bei der zukünftigen Verschmelzung von Multimedia mit modernen Kommunikationstechnologien wird dort ein gewaltiges Marktwachstum zu erwarten sein.

Von einem *Anwenderstandpunkt* aus gesehen, der die Bionik als Transfermechanismus materieller Produkte oder "Ideen" des Lebens (z.B. Delphinhaut) in die Technik (z.B. Schiffsaußenhülle) betrachtet, könnte man AL als Transfer "logischer Formen" des Lebens (z.B. Schwarm*verhalten*) in die Technik (z.B. Verkehrsleitsystem) auffassen.

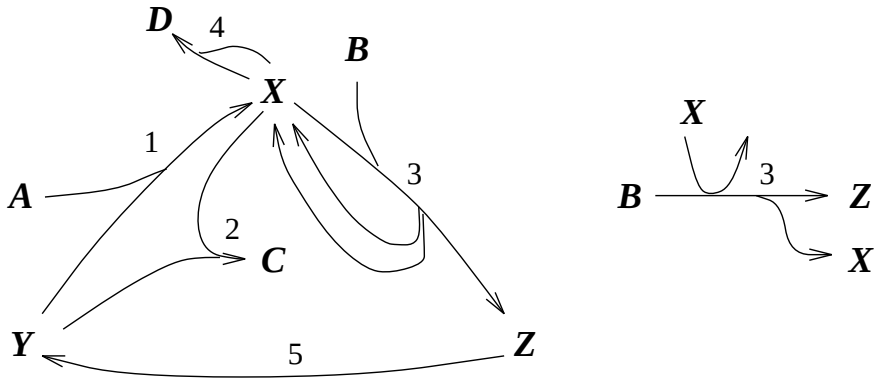


Abb. 1.1: *Links: Reaktionsgraph zum Beispiel 1.3.1. Rechts: Eine alternative Darstellung der dritten Reaktion. Dabei wird die katalytische Wirkung von X deutlicher. In diesem speziellen Fall ist die Reaktion sogar auto-katalytisch, da der Katalysator selbst erzeugt wird.*

1.3 Reaktionssysteme

In dieser Arbeit werden algorithmische Systeme untersucht, die ihr Analogon auf dem molekularen Level lebender Organismen haben (vgl. Abschnitt 1.5). Es werden sogenannte Reaktionssysteme betrachtet. Ein **Reaktionssystem** besteht aus einer Menge von Stoffen und aus Regeln, die die Wechselwirkungen zwischen den Stoffen beschreiben. Diese Regeln bestimmen, wie die Stoffe untereinander reagieren, um so andere bzw. neue Stoffe zu bilden. Die Formulierung der Regeln kann durch **Reaktionsgleichungen** erfolgen, die beschreiben, welche Stoffe in welchen Mengenverhältnissen in andere Stoffe umgesetzt werden.

Beispiel 1.3.1 *Ein chemisches Reaktionssystem mit den Stoffen A, B, C, D, X, Y, Z und den Reaktionsgleichungen:*

$$\begin{array}{lrl} 1. & A + Y & \longrightarrow X \\ 2. & X + Y & \longrightarrow C \\ 3. & B + X & \longrightarrow 2X + Z \\ 4. & 2X & \longrightarrow D \\ 5. & Z & \longrightarrow Y \end{array} \qquad (1.1)$$

(aus [Atkins 1988], S. 745)

Die dritte Reaktionsgleichung (1.1) besagt zum Beispiel, daß je eine Mengeneinheit des Stoffes B und X zu zwei Mengeneinheiten X und einer Einheit Z umgesetzt werden können.

Zur Visualisierung der Zusammenhänge zwischen den Reaktionsgleichungen dient ein **Reaktionsgraph** (Abb. 1.1).

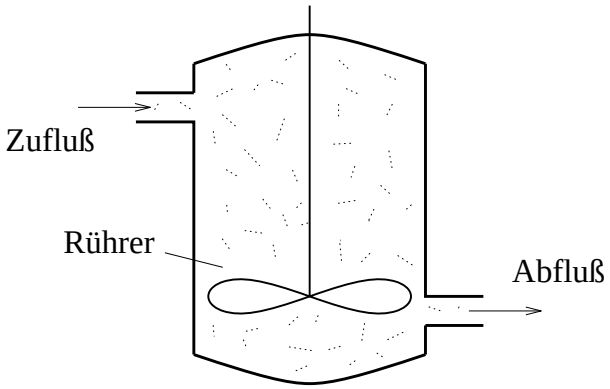


Abb. 1.2: *Ein kontinuierlich arbeitender gerührter Tankreaktor. Es handelt sich dabei um das technische Analogon zum Reaktor-Algorithmus.*

1.3.1 Der allgemeine Reaktor

In der großtechnischen Chemie erfolgt die Umsetzung von Stoffen in sogenannten **chemischen Reaktoren**. Allgemein soll unter einem **Reaktor** ein konkretes oder abstraktes Reaktionsgefäß verstanden werden, in dem Reaktionen zwischen **Stoffen** stattfinden. Während des Betriebs können kontinuierlich oder diskontinuierlich Stoffe zugegeben oder abgelassen werden. Die Stoffentnahme kann im allgemeinen nicht selektiv erfolgen. Vielmehr ist der **Abfluß** einer Stoffmenge proportional zu ihrer Konzentration. Üblicherweise wird der Fluß so eingestellt, daß die Stoffmenge im Reaktor konstant bleibt.

Bei einem **chemischen Reaktor** unterscheidet Atkins ([Atkins 1988], S. 718) zwischen einem :

1. **kontinuierlich arbeitenden gerührten Tank** (engl. continuous stirred tank reactor, CSTR) mit kontinuierlicher Zufuhr der Ausgangssubstanzen und kontinuierlicher Abfuhr der Produkte (Abb. 1.2).
2. **Röhrenreaktor**, bei dem ein stationärer Fluß von Material durch ein Rohr fließt, in dem die Reaktionen stattfinden.
3. **diskontinuierlich betriebenen Kessel**.

Die im folgenden betrachteten formalen Reaktoren entsprechen im wesentlichen einem kontinuierlich arbeitenden gerührten Tank. Die wichtige Eigenschaft eines solchen Reaktors besteht darin, daß sein Inhalt räumlich **homogen** ist, also die Konzentration eines Stoffes unabhängig vom Ort im Reaktor ist.

1.3.2 Dynamik eines Reaktionssystems

Das Gleichungssystem von Beispiel 1.3.1 sagt nichts über die Dynamik des Reaktionssystems aus. Die Art und Weise wie chemische Reaktionen ablaufen ist äußerst vielfältig (z.B. rosten und verbrennen) und mathematisch schwer zu erfassen. Einfache formale Modelle sind meist nur für bestimmte Situationen anwendbar (z.B. Gasphase oder wässrige

Lösung). Die einfachste Möglichkeit, das dynamische Verhalten zu quantifizieren, erfolgt mit Hilfe einer skalaren **Geschwindigkeitskonstante** k_i für jede Reaktionsgleichung i. Man nimmt dabei an, daß die umgesetzte Stoffmenge in einem kleinen Zeitintervall proportional zu den Konzentrationen der Ausgangsstoffe ist. Dieser Sachverhalt kann als ein System gewöhnlicher Differentialgleichungen formuliert werden.

Beispiel 1.3.2 *Für das Beispiel 1.3.1 ergibt sich dann folgendes DGL-System:*

$$\begin{aligned}
d[A]/dt &= -k_1[A][Y] \\
d[B]/dt &= -k_3[B][X] \\
d[C]/dt &= k_2[X][Y] \\
d[D]/dt &= k_4[X]^2 \\
d[X]/dt &= k_1[A][Y] - k_2[X][Y] + k_3[B][X] - k_4[X]^2 \\
d[Y]/dt &= -k_2[X][Y] + k_5[Z] \\
d[Z]/dt &= k_3[B][X] - k_5[Z]
\end{aligned} \tag{1.2}$$

Die letzte Gleichung (1.2) des DGL-Systems kann folgendermaßen gelesen werden: Die Konzentration $[Z]$ des Stoffes Z wird wegen der Reaktionsgleichung $B + X \longrightarrow 2X + Z$ erhöht und zwar proportional zu den Konzentrationen von B und X mit der Proportionalitätskonstante k_3. Dagegen wirkt eine Abnahme des Stoffes wegen der fünften Reaktion $Z \longrightarrow Y$ mit der Geschwindigkeit k_5.

Für ein realistisches Modell können die k_i nicht als konstant angenommen werden, sondern hängen von Einflüssen wie Temperatur, Druck, Beleuchtung, Ionenkonzentrationen (z.B. pH-Wert) usw. ab, die wiederum vom System selbst beeinflußt werden können. Zum Beispiel erzeugt oder verbraucht praktisch jede Reaktion Wärme. Ferner ist die Eigenschaft der Homogenität nur selten gesichert, so daß zur genaueren Beschreibung partielle DGLs eingeführt werden müßten.

Hier werden aber ausschließlich homogene Systeme untersucht. Die Einführung räumlicher Strukturen ist anderen Arbeiten überlassen [Eller 1995].

1.4 Populationsdynamik

Auf der Populationsebene der Fauna und Flora finden sich ähnliche dynamische Phänomene wie auch in chemischen Reaktoren. Die formalen Modelle, die die theoretische Biologie zu ihrer Beschreibung verwendet, sind denen für chemische Reaktoren sehr ähnlich und zum Teil identisch.

Beispiel 1.4.1 *Das bekannte* **Räuber-Beute-Modell** *nach* Lotka *und* Volterra *kann in Form von Reaktionsgleichungen notiert werden:*

$$A + X \longrightarrow 2X \quad \text{"Beute } X \text{ frißt Grundnahrung } A \text{ und vermehrt sich."} \tag{1.3}$$

$$X + Y \longrightarrow 2Y \quad \text{"Räuber } Y \text{ frißt Beute } X \text{ und vermehrt sich."} \tag{1.4}$$

$$Y \longrightarrow B \quad \text{"Räuber stirbt."} \tag{1.5}$$

Ist die Versorgung der Grundnahrung A für die Beute konstant, so ergibt sich das DGL-System:

$$d[X]/dt = k_1[A][X] - k_2[X][Y] \qquad \text{"Beute"} \tag{1.6}$$

$$d[Y]/dt = k_2[X][Y] - k_3[Y] \qquad \text{"Räuber"} \tag{1.7}$$

Auch andere theoretische Gebiete der Biologie, wie die Populationsgenetik, Evolutionstheorie und Soziobiologie, verwenden Modelle obiger Art. Für eine ausführliche Darstellung sei hier auf das Lehrbuch von HOFBAUER und SIGMUND verwiesen [HOFBAUER UND SIGMUND 1984].

1.5 Genetik: Protein-Biosynthese

Der genetische Apparat einer Zelle besteht *stark vereinfacht* aus Nukleinsäuren (DNS und RNS) und Proteinen [LÖFFLER UND PETRIDES 1984]. Die **Nukleinsäuren** sind lange Ketten, die aus vier **Nukleotiden** gebildet werden. **Proteine** sind lange Sequenzen aus **Aminosäuren**, von denen es ca. 20 verschiedene gibt. Eine **DNS** kann man sich als gut geschütztes Masterband vorstellen, das als Muster zur Produktion von RNS dient (**Transkription**). **RNS** ist ähnlich wie DNS aufgebaut, allerdings kürzer und instabiler, da sie nicht die robuste Doppelhelixstruktur der DNS annehmen. RNS wiederum dient zur Produktion der Proteine (**Translation**), indem nach einem festen **genetischen Code** drei Nukleotide in eine Aminosäure übersetzt werden. Zur Durchführung bzw. Katalyse der Transkription und Translation werden bestimmte Proteine verwendet. Auch sind Proteine zur identischen Replikation der DNS unabdingbar.

Im Rahmen des Komplexes "Entstehung des Lebens" stellt sich nun die Frage, gab es zuerst Proteine (Katalysatoren, "Maschinen") oder zuerst Nukleinsäuren (Muster, "Daten").

Eine Antwort könnten neuere Erkenntnisse der Molekularbiologie liefern, wonach RNS nicht nur als Datenstrang (Operand) dient, der von den "Proteinmaschinen" verarbeitet wird, sondern auch selbst in gefalteter Form als Katalysator (Operator) fungieren kann [KRUGER UND ET AL. 1982; FRAUENFELDER 1988]. Diese Entdeckung spricht für die Annahme, daß eine selbsterhaltende "RNS-Welt" unabhängig von Proteinen existiert haben könnte.

Die Operator-Operanden-Dualität der RNS-Welt findet sich auch in den untersuchten formalen (algorithmischen) Reaktionssystemen. Alle Binärstrings können dort sowohl als Daten als auch als Operatoren auftreten, die andere Binärstrings verarbeiten können.

Kapitel 2

Formale Reaktionssysteme

Ein **formales Reaktionssystem** ist ein System, welches das Verhalten oder die Funktionsweise chemischer Reaktionen nachahmt. Ein **algorithmisches Reaktionssystem** ist ein spezielles formales Reaktionssystem, das als Algorithmus notiert werden kann. Der Unterschied zu einem allgemeinen formalen System besteht darin, daß "nur" endlich viel Zeit und ein beliebig großer, aber endlicher Speicher zur Verfügung stehen.

Zum Beispiel kann ein allgemeines DGL-System nicht als algorithmisch bezeichnet werden, da zu seiner Simulation auf einer Turing-Maschine sowohl die Zeit als auch der Zustandsraum diskretisiert werden müssen. Das Verhalten der entsprechenden Turing-Maschine ist nur eine Näherung für die Dynamik des DGL-Systems. Ein algorithmisches Reaktionssystem kann dagegen *exakt* durch eine Turing-Maschine simuliert werden. Man spricht deshalb besser von einer Realisierung, Instanziierung oder Implementation des Reaktionssystems.

Im Mittelpunkt dieser Arbeit steht ein algorithmisches Reaktionssystem, das durch den folgenden Reaktor-Algorithmus beschrieben wird.

2.1 Der Reaktor-Algorithmus

Der **Reaktor-Algorithmus** arbeitet auf einer Population P von M Objekten aus einer Menge S. Eine einfache Basisversion, die in der Versuchsreihe A (vgl. Kapitel 4) eingesetzt wird, lautet:

Algorithmus 2.1.1 *Einfacher Reaktor-Algorithmus mit konstanter Produktionsrate:*

- *Initialisierung (Befüllung des Reaktors):*
  Erzeuge M zufällige Objekte. Diese bilden die Population P.
- *Iteration (Betrieb des Reaktors):*
  1. *Wähle zufällig zwei Objekte $\mathbf{s}, \mathbf{s}'$ aus der Population aus, ohne diese zu entfernen.*
  2. *Lasse $\mathbf{s}, \mathbf{s}'$ miteinander zu $\mathbf{s}'' := \mathbf{s} \circ \mathbf{s}'$ reagieren.*
  3. *Ersetze ein zufälliges Objekt der Population durch $\mathbf{s}''$.*

Bei dieser einfachen Variante wird in jeder Iteration ein Objekt der Population durch das Produkt einer Reaktion ersetzt. Es herrscht ein konstanter Abfluß Φ und Zufluß von einem Objekt pro Iteration. Die Populationsgröße bleibt konstant.

In der allgemeinen Version sollen Interaktionen zwischen Objekten **elastisch** sein können. Das heißt, daß beim Zusammentreffen bestimmter Objekte keine Reaktion stattfindet. Der Algorithmus, der dies berücksichtigt, lautet:

Algorithmus 2.1.2 Reaktor-Algorithmus*:*

- *Initialisierung (Befüllung des Reaktors):*
  Erzeuge M zufällige Objekte aus S. Diese bilden die Population P.
- *Iteration (Betrieb des Reaktors):*
  1. *Wähle zufällig zwei Objekte $\mathbf{s}, \mathbf{s}'$ aus der Population aus, ohne diese zu entfernen.*
  2. *Falls $\mathbf{s}, \mathbf{s}'$ miteinander reagieren, erzeuge ihr Reaktionsprodukt $\mathbf{s}'' := \mathbf{s} \circ \mathbf{s}'$.*
  3. *Haben $\mathbf{s}, \mathbf{s}'$ miteinander reagiert, so ersetze zufällig ein Objekt der Population durch $\mathbf{s}''$.*

Diese allgemeine Version wird in den Versuchsreihen B und C (vgl. Kapitel 4) verwendet. Dabei ist die Bedingung für eine reaktive Wechselwirkung zwischen $\mathbf{s}$ und $\mathbf{s}'$ die, daß das Reaktionsprodukt $\mathbf{s}''$ weder gleich $\mathbf{s}$ noch $\mathbf{s}'$ sein darf. Eine solche Bedingung wird **Replikationsverbot** genannt.

Die Laufzeit kann entweder in "Iterationen" oder in "Generationen" angegeben werden. Unter einer **Generation** versteht man M Iterationen. Die **Produktionsrate** zu einem Zeitpunkt ist die Wahrscheinlichkeit, mit der während einer Iteration ein Reaktionsprodukt $\mathbf{s}''$ entsteht. Für die Darstellung der Produktionsrate wird hier ein Schätzwert verwendet, der sich aus der Anzahl der reaktiven Iterationsschritte in einer Generation geteilt durch die Populationsgröße M ergibt.

Die durch den Algorithmus erzeugte Dynamik ist der chemischer Reaktionen mit dem Reaktionstyp

$$A + \mathbf{s} + \mathbf{s}' \longrightarrow \mathbf{s} + \mathbf{s}' + \mathbf{s}'' \tag{2.1}$$

ähnlich. Das **Substrat** A taucht im Algorithmus nicht auf, und seine Konzentration wird für die Aufstellung eines DGL-Systems als konstant angenommen. Das Substrat ist in chemischen und biochemischen Systemen der Energielieferant. Frei nach T. S. Ray [Ray 1991], könnte man A mit der Rechenzeit eines Computers gleichsetzen, auf dem der Reaktor-Algorithmus läuft. Zu der Gleichung 2.1 ist folgende Schreibweise äquivalent:

$$\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$$

Um die Konzentrationsverläufe mit gewöhnlichen DGLs beschreiben zu können, müssen zwei vereinfachende Annahmen gemacht werden:

1. **Die Population ist groß.**
   Das heißt, daß jedes Objekt in einer großen Anzahl in der Population vertreten ist, so daß sich mikroskopisches Verhalten in makroskopischen Konzentrationsänderungen akkumuliert und stochastische Effekte vernachlässigt werden können. Ohne diese Annahme müßte man stochastische DGLs oder eine sogenannte "Metadynamik" betrachten [FARMER, KAUFFMAN, UND PACKARD 1986].

2. **Die Population ist räumlich homogen.**
   Damit ist keine Modellierung räumlicher Konzentrationsverteilungen nötig. Gewöhnliche DGLs sind ausreichend.

Unter diesen Annahmen kann der Zustand des Reaktors vollständig durch einen Punkt $\mathbf{x} = (x_1, \ldots, x_n)$ im **Einheits-Konzentrationssimplex**

$$\mathrm{S}_n = \{\mathbf{x} \in \mathbb{R}^n : \sum_{i=1}^{n} \mathrm{x_i} = 1, \mathrm{x_i} > 0\} \tag{2.2}$$

bestimmt werden. x_i beschreibt die Konzentration des Stoffes $\mathbf{s}^{(i)}$.

Beispiel 2.1.1 Reaktionssystem mit zwei Spezies $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}$:
Reaktionsgleichungen:

$$\begin{array}{lcl}
A + \mathbf{s}^{(1)} + \mathbf{s}^{(1)} \longrightarrow \mathbf{s}^{(1)} + \mathbf{s}^{(2)} + \mathbf{s}^{(2)} & & \mathbf{s}^{(1)} + \mathbf{s}^{(1)} \Longrightarrow \mathbf{s}^{(2)} \\
A + \mathbf{s}^{(1)} + \mathbf{s}^{(2)} \longrightarrow \mathbf{s}^{(1)} + \mathbf{s}^{(2)} + \mathbf{s}^{(2)} & & \mathbf{s}^{(1)} + \mathbf{s}^{(2)} \Longrightarrow \mathbf{s}^{(2)} \\
A + \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \longrightarrow \mathbf{s}^{(1)} + \mathbf{s}^{(2)} + \mathbf{s}^{(2)} & \textit{oder} & \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \Longrightarrow \mathbf{s}^{(2)} \\
A + \mathbf{s}^{(2)} + \mathbf{s}^{(2)} \longrightarrow \mathbf{s}^{(1)} + \mathbf{s}^{(2)} + \mathbf{s}^{(1)} & & \mathbf{s}^{(2)} + \mathbf{s}^{(2)} \Longrightarrow \mathbf{s}^{(1)}
\end{array}$$

Reaktionsmatrix:

	$\mathbf{s}^{(1)}$	$\mathbf{s}^{(2)}$
$\mathbf{s}^{(1)}$	$\mathbf{s}^{(2)}$	$\mathbf{s}^{(2)}$
$\mathbf{s}^{(2)}$	$\mathbf{s}^{(2)}$	$\mathbf{s}^{(1)}$

oder kürzer

	1	2
1	2	2
2	2	1

Das zugehörige DGL-System lautet:

$$\begin{aligned}
\frac{dx_1}{dt} &= x_2 x_2 - x_1 \\
\frac{dx_2}{dt} &= x_1 x_1 + x_1 x_2 + x_2 x_1 - x_2 \qquad \textit{mit} \quad (x_1, x_2) \in \mathrm{S}_2
\end{aligned} \tag{2.3}$$

Die Abbildung 2.1 zeigt einen Vergleich des Reaktor-Algorithmus mit dem DGL-System.

Falls mehr als zwei Spezies nebeneinander vorliegen (also für $n > 2$), ist das DGL-System im allgemeinen nicht analytisch lösbar [HOFBAUER UND SIGMUND 1984].

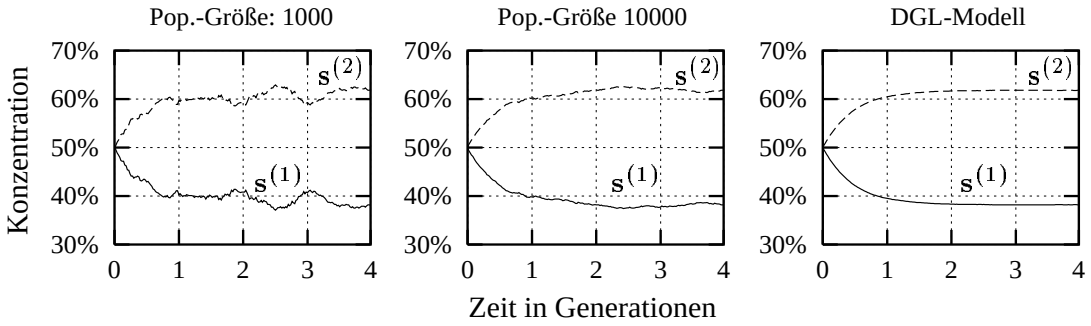


Abb. 2.1: *Vergleich zwischen dem Reaktor-Algorithmus und dem korrespondierenden DGL-Modell für das Reaktionssystem im Beispiel 2.1.1. Je kleiner die Population wird, desto stärker machen sich stochastische Effekte bemerkbar. Das DGL-Modell abstrahiert von den nicht-deterministischen Schwankungen und kann als Grenzwert für den Fall $M \to \infty$ oder als Überlagerung unendlich vieler Runs mit endlicher Populationsgröße aufgefaßt werden. Links: Reaktor-Algorithmus mit einer Populationsgröße von 1000 Objekten. Mitte: Reaktor-Algorithmus mit einer Populationsgröße von 10000 Objekten. Rechts: DGL-Modell (2.3).*

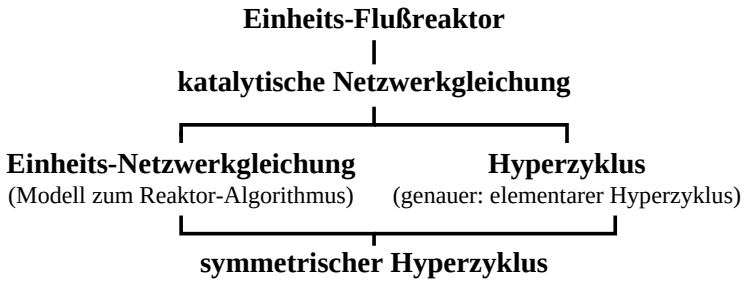


Abb. 2.2: *Hierachie der vorgestellten DGL-Modelle für räumlich homogene Reaktionssysteme. Der Einheits-Flußreaktor ist die allgemeinste Form. Alle Systeme haben eine konstante Populationsgröße und einen nicht-selektiven Abfluß der Substanzen gemeinsam. Das heißt, daß der Anteil einer Substanz im Abfluß proportional zu ihrer Konzentration ist.*

2.2 Gewöhnliche Differentialgleichungen

Nachdem in den Beispielen schon eine Reihe konkreter DGL-Modelle vorgestellt worden sind, soll nun ein Überblick über DGL-Modelle für räumlich homogene Reaktionssysteme gegeben werden. Zunächst wird ein allgemeines Modell vorgestellt – der Einheits-Flußreaktor. Davon ausgehend werden dann interessante Spezialfälle abgeleitet (Abb. 2.2).

2.2.1 Der Einheits-Flußreaktor

Der Zustand des **Einheits-Flußreaktors** ist vollständig durch einen Punkt $\mathbf{x} = (x_1, \ldots, x_n)$ im **Einheits-Konzentrationssimplex**

$$\mathrm{S}_n = \{x \in \mathbb{R}^{\mathrm{n}} : \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i} = 1, \mathrm{x_i} > 0\}$$

beschrieben. x_i ist die Konzentration des Stoffes $\mathbf{s}^{(i)}$. Zur Vereinfachung kann für $\mathbf{s}^{(i)}$ einfach i geschrieben werden. Durch einen **Verdünnungsfluß** Φ wird die **Gesamtkonzentration**

$$S_x = x_1 + x_2 + \cdots + x_n \equiv 1$$

konstant gleich eins gehalten. Die Dynamik ist durch ein gekoppeltes System gewöhnlicher Differentialgleichungen gegeben ([Hofbauer und Sigmund 1984] S. 118 ff.):

$$\boxed{\frac{dx_k}{dt} = \Gamma_k(x_1, \ldots, x_n) - x_k\Phi(x_1, \ldots, x_n), \qquad k = 1, \ldots, n} \tag{2.4}$$

$\Gamma_k(x_1, \ldots, x_n)$ heißt **Wachstumsterm** und beschreibt den Effekt der Wechselwirkung. Der **Verdünnungsterm** $-x_k\Phi(x_1, \ldots, x_n)$ besagt, daß der Anteil der Substanz $\mathbf{s}^{(k)}$ im Verdünnungsfluß Φ proportional zu ihrer Konzentration x_k ist.

Aus der Bedingung $S_x \equiv 1$ folgt für den Verdünnungsfluß Φ:

$$\Phi(x_1, \ldots, x_n) = \sum_{i=1}^{n} \Gamma_i(x_1, \ldots, x_n)$$

2.2.2 Die katalytische Netzwerkgleichung

Beschränkt man sich darauf, daß der Wachstumsterm des Einheits-Flußreaktors quadratisch in den Konzentrationen x_i ist, so erhält man die **katalytische Netzwerkgleichung** (engl. **katalytic network equation**) [Stadler, Fontana, und Miller 1991]. Sie ist ein Spezialfall der Gleichung (2.4), aber dennoch ein äußerst allgemeines Modell, das viele spezielle Modelle der präbiotischen Evolution und Populationsgenetik umfaßt. Die Katalytische Netzwerkgleichung lautet:

$$\boxed{\frac{dx_k}{dt} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}^k x_i x_j - x_k\Phi(t), \qquad k = 1, \ldots, n} \tag{2.5}$$

mit der **Geschwindigkeitskonstante** (engl. rate constant, [Atkins 1988] S. 719) zweiter Ordnung α_{ij}^k für die Reaktion

$$A + i + j \xrightarrow{\alpha_{ij}^k} i + j + k \qquad \text{oder kurz} \qquad i + j \overset{\alpha_{ij}^k}{\Longrightarrow} k$$

und dem **Abfluß** (engl. **dilution flux**) $\Phi(t)$, so daß $S_x = \sum_k x_k(t) \equiv 1$ ist. Die Reaktionsgleichung kann folgendermaßen gelesen werden: Unter der katalytischen Wirkung von

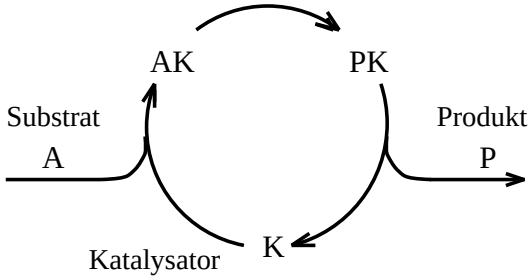


Abb. 2.3: *Darstellung der Wirkung eines Katalysators K als Reaktionszyklus. (Nach [*Eigen und Schuster 1977*] Abb. 6; s. auch Stichwort "Michaelis-Menten-Gleichung" [*Voet und Voet 1992*] S. 329).*

i und j wird das energiereiche **Substrat** A zu k umgesetzt. Es wird davon ausgegangen, daß das Substrat ständig in ausreichender Menge verfügbar ist. In der obigen Reaktionsgleichung taucht es nur zur Erhaltung der Stoffbilanz auf und läßt sich abstrahieren. α_{ij}^k kann interpretiert werden als Anteil der reaktiven Kollisionen zwischen i und j, die die Entstehung von k zufolge haben.

Für den Verdünnungsfluß $\Phi(t)$ folgt aus der Bedingung $S_x \equiv 1$:

$$\Phi(t) = \sum_{i,j,k=1}^{n} \alpha_{ij}^k x_i x_j$$

2.2.3 Die Einheits-Netzwerkgleichung zum Reaktor-Algorithmus

Der Reaktor-Algorithmus 2.1.2 bevorzugt keine Reaktion. Jede Reaktion läuft mit der gleichen Geschwindigkeit ab. Deshalb kann o. B. d. A. die Reaktionsgeschwindigkeit für jede Reaktion gleich 1 gesetzt werden.

$$\boxed{\alpha_{ij}^k = \begin{cases} 1 & \text{falls } i \text{ mit } j \text{ zu } k \text{ reagiert} \\ 0 & \text{sonst} \end{cases}} \tag{2.6}$$

Um den Reaktor-Algorithmus derartig modellieren zu können, muß vorausgesetzt werden, daß 1.) die Reaktion von i mit j eindeutig (deterministisch) ist, 2.) das System abgeschlossen bezüglich dieser Reaktion ist (also: $\forall i, j \in S : i \circ j \in S_p \subseteq S$) und 3.) S_p endlich ist.

Die Bedingung der Abgeschlossenheit und Endlichkeit der Menge S bzw. S_p gilt zwar für diese Arbeit, allerdings nicht allgemein für algorithmische Reaktionssysteme. Mit dem Wunsch nach einer "Evolution mit offenem Ende" (engl. open ended evolution) wird die Offenheit des Systems sogar angestrebt. Dies wird dann ermöglicht, falls die Größe des Speicherplatzes für ein Objekt variabel ist und beliebig groß werden kann.

2.2.4 Der elementare Hyperzyklus

Reaktionszyklen wie im Beispiel 1.3.1 tauchen häufig in der Natur auf. Der von Bethe und von Weizäcker 1938 vorgeschlagene **Carbonzyklus** [Bethe 1969] ist verantwortlich

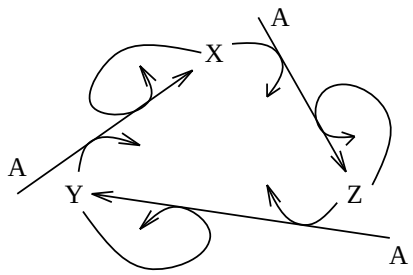


Abb. 2.4: *Beispiel für einen katalytischen Hyperzyklus. Der Hyperzyklus wird von drei Spezies gebildet, die über die autokatalytischen Reaktionsgleichungen* $X + Z \Longrightarrow Z$, $Z + Y \Longrightarrow Y$ *und* $Y + X \Longrightarrow X$ *gekoppelt sind. Es handelt sich dabei um Reaktionen zweiter Ordnung bezüglich der Katalysatoren. Die Spezies haben zwei Tätigkeitsfelder. Sie unterstützen ihre eigene Produktion und die einer anderen Spezies. (Nach [*Eigen und Schuster 1977*] Abb. 7).*

für die Energieproduktion in unserer Sonne. Im menschlichen Körper ist der **Zitratzyklus**, der (ebenfalls um 1938) von Krebs enddeckt wurde, wichtig zur Energiegewinnung ([Krebs 1964], [Löffler und Petrides 1984] S. 339 ff.). Auch die Wirkung eines Katalysators kann man als einen einfachen Reaktionszyklus darstellen (Abb. 2.3).

Der Hyperzyklus ist ein spezieller Reaktionszyklus, der von Eigen und Schuster Anfang der 70er Jahre vorgeschlagen wurde [Eigen und Schuster 1977; Eigen und Schuster 1978a; Eigen und Schuster 1978b; Eigen, Gardiner, Schuster, und Winkler-Oswatitsch 1981; Eigen und Schuster 1982; May 1991]. Es handelt sich dabei um eine Organisationsform, die die Entstehung des genetischen Apparates bzw. den Ursprung des Lebens klären soll. Ein **Hyperzyklus** (genauer: katalytischer Hyperzyklus) ist ein System, das aus autokatalytischen bzw. replikativen Einheiten besteht, die durch einen Zyklus miteinander verbunden sind (Abb. 2.4).

Ein Reaktionszyklus, wie der Carbonzyklus, kann als ganzes wie ein Katalysator behandelt werden, der die Umsetzung von Wasserstoff in Helium beschleunigt. Eine zyklische Anordnung verschiedener derartiger Zyklen – also ein Zyklus von Zyklen – ist im Sinne von Eigen und Schuster *kein* Hyperzyklus. Der Begriff **Hyperzyklus** wird für Systeme verwendet, die hyperzyklisch im Sinne der katalytischen Funktionen sind. Dabei müssen die Reaktionen mindestens von zweiter Ordnung bezüglich der Katalysatoren sein. Dies trift im Fall des Reaktor-Algorithmus und der katalytischen Netzwerkgleichung zu. In einem hyperzyklischen System werden die Katalysatoren durch den Zyklus selbst produziert. Im Prinzip sind die Spezies für zwei Aufgaben verantwortlich: 1.) für ihre eigene Replikation und 2.) für die Unterstützung der Replikation einer andern Spezies (vgl. [Eigen und Schuster 1977] S. 545).

Einen Hyperzyklus erhält man aus der katalytischen Netzwerkgleichung, falls man

$$\alpha_{ij}^{k} = \begin{cases} \alpha_k & \text{falls} \quad i = k, j = k-1 \text{ und } k > 1, \\ \alpha_1 & \text{falls} \quad i = k = 1 \text{ und } j = n, \\ 0 & \text{falls} \quad \text{sonst} \end{cases}$$

setzt mit nicht verschwindenden $\alpha_k \in \mathbb{R}^+$. Die Hyperzyklusgleichung lautet also für $k > 1$:

$$\frac{dx_k}{dt} = \alpha_k x_k x_{k-1} - x_k \Phi$$

Und um den Kreis zu schließen für $k = 1$:

$$\frac{dx_1}{dt} = \alpha_1 x_1 x_n - x_1 \Phi$$

Für eine einfachere Darstellung definiert man $x_0 := x_n$. Die **Hyperzyklusgleichung** kann dann geschrieben werden als:

$$\boxed{\frac{dx_k}{dt} = x_k(\alpha_k x_{k-1} - \sum_{i=1}^{n} \alpha_k x_i x_{i-1})} \tag{2.7}$$

Diese Gleichung beschreibt ein Reaktionsnetzwerk mit nur wenigen reaktiven Interaktionen. Kollisionen zwischen zum Beispiel $\mathbf{s}_2$ und $\mathbf{s}_6$ sind elastisch. Deshalb spricht man auch genauer bei Gleichung (2.7) von einem **elementaren Hyperzyklus** (engl. elementary hypercycle, [Eigen und Schuster 1978a] S. 23).

2.2.5 Der symmetrische Hyperzyklus

Setzt man – wie bei der Einheits-Netzwerkgleichung – die Konstanten $\alpha_k = 1$ ($k = 1, \ldots, n$), so erhält man die **symmetrische Hyperzyklus-Gleichung**:

$$\boxed{\frac{dx_k}{dt} = x_k(x_{k-1} - \sum_{i=1}^{n} x_k x_{k-1})} \tag{2.8}$$

Mit Hilfe der **baryzentrischen Koordinatentransformation** ([Hofbauer und Sigmund 1984] S. 127) lassen sich die Koordinaten eines Hyperzyklus nach Gleichung (2.7) auf einen symmetrischen (2.8) transformieren. Da diese Transformation mitsamt ihrer Umkehrung differenzierbar ist, bleibt die qualitative topologische Eigenschaft der stationären Zustände erhalten, was direkt aus der Kettenregel folgt.

2.3 Konkrete algorithmische Reaktionssysteme

In dieser Arbeit werden Reaktionssysteme untersucht, bei denen die Objekte aus einer Menge $S \subseteq \mathbb{B}^*$ stammen, wobei $\mathbb{B}$ genau zwei Elemente entält. Worte $\mathbf{s}$ über der Menge $\mathbb{B}$ heißen **Binärstrings**. O. b. d. A. kann definiert werden: $\mathbb{B} = \{0, 1\}$.

In [Banzhaf 1993a; Banzhaf 1993b; Banzhaf 1994; Banzhaf 1995] werden Binärstring-Reaktionssysteme vorgeschlagen, bei denen Binärstrings in zwei verschiedenen Formen auftreten; 1.) als Daten und 2.) als Operatoren, die andere Strings als Operanden verarbeiten und wiederum Strings als Ausgabe liefern. Der Übergang eines Binärstrings in seine Operatorform wird analog zur Biochemie **Faltung** genannt. Der Operator hat bei Banzhaf die Form einer zwei-dimensionalen Matrix.

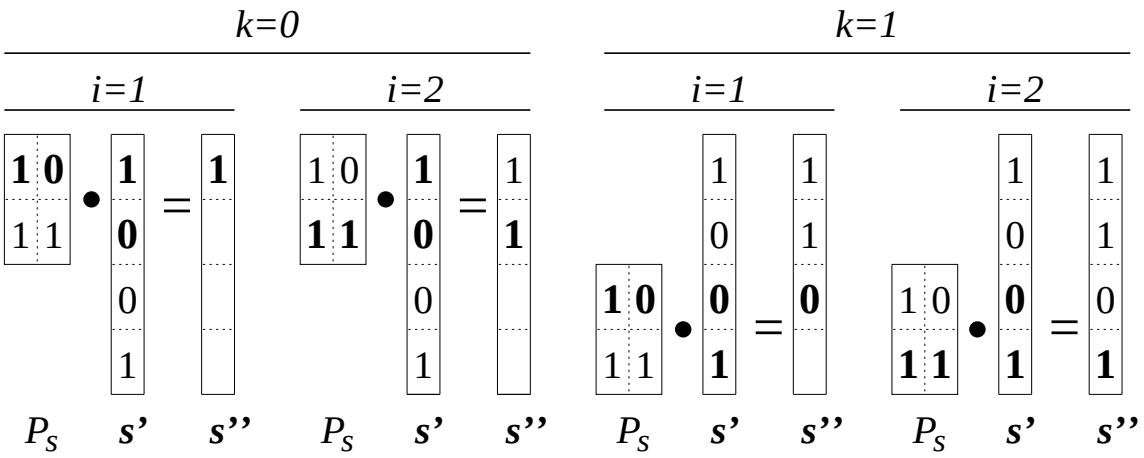


Abb. 2.5: *Die Arbeitsweise eines Operators nach Gleichung 2.11 mit $\theta = 1$. Der Operator ist aus der Faltung des Strings* $\mathbf{s} = (1,0,1,1)$ *hervorgegangen. Durch die Anwendung auf den Operanden-String* $\mathbf{s}' = (1,0,0,1)$ *entsteht das Reaktionsprodukt* $\mathbf{s}'' = (1,1,0,1)$.

2.3.1 Ein Binärstring-Reaktionssystem nach BANZHAF

Zunächst werden Binärstrings von konstanter Länge N betrachtet mit $\sqrt{N} \in \mathbb{N}$. Die Faltung eines Binärstrings $\mathbf{s} = (s_1, s_2, \ldots, s_N)$ mit $s_i \in \{0,1\}$ zu einer quadratischen Matrix P_s kann für den Fall $N = 4$ bzw. $N = 9$ folgendermaßen definiert werden:

$$\mathbf{s} = (s_1, s_2, s_3, s_4) \longrightarrow \mathrm{P}_s = \begin{pmatrix} s_1 & s_2 \\ s_4 & s_3 \end{pmatrix} \tag{2.9}$$

$$\mathbf{s} = (s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9) \longrightarrow \mathrm{P}_s = \begin{pmatrix} s_1 & s_2 & s_3 \\ s_6, & s_5 & s_4 \\ s_7 & s_8 & s_9 \end{pmatrix} \tag{2.10}$$

Für die Reaktion $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ wird zunächst $\mathbf{s}$ zu einer Matrix P_s gefaltet. Dann wird das Skalarprodukt verwendet, um sukzessiv Teilstrings der Größe $\sqrt{N}$ von $\mathbf{s}'$ auf die entsprechenden Teilstrings von $\mathbf{s}''$ abzubilden. Durch das Skalarprodukt können Werte größer 1 entstehen. Deshalb wird ein Grenzwert θ eingeführt, so daß Werte kleiner θ auf 0 und sonst auf 1 gesetzt werden. Die Abbildung 2.5 zeigt ein Beispiel für $N = 4$ und $\theta = 1$. Allgemein gilt:

$$\mathbf{s}''_{i+k\sqrt{N}} = \sigma\left[\sum_{j=1}^{\sqrt{N}} (\mathrm{P}_s)_{ij}\, s'_{i+k\sqrt{N}} - \theta\right], \quad i = 1, \ldots, \sqrt{N}, \quad k = 0, \ldots, \sqrt{N} - 1 \tag{2.11}$$

mit

$$\sigma(x) = \begin{cases} 0 & \text{falls } x < 0, \\ 1 & \text{sonst} \end{cases}$$

In diesem System existieren sogenannte **lethale Spezies**. Ein String heißt **lethal**, falls er überproportional oft zur Replikation fähig ist ([BANZHAF 1993B] S. 272). Der String

$\mathbf{s}^{(0)} = (0,0,0,0)$ repliziert sich mit jedem anderen String. Innerhalb einer Population verdrängt er vollständig alle anderen Spezies und wird deshalb **Destruktor** genannt. Um seine zerstörerische Wirkung zu verhindern, wird er einfach verboten. Tritt er in der Population auf, so wird er durch einen zufälligen String ersetzt (s. Alg. 2.3.1, Schritt 5). Ein weiterer lethaler String ist der sogenannte **Exploiter** $\mathbf{s}^{(15)} = (1,1,1,1)$, der nicht ganz so "egoistisch" wie der Destruktor ist, aber sich dennoch mit Hilfe vieler Spezies replizieren kann. Da er nicht grundsätzlich alle anderen Strings verdrängt, sind auch keine derartig drastische Maßnahmen wie gegen den Destruktor notwendig. Es wird ein Selektionsdruck eingeführt, durch den Strings mit wenig Einsen bevorzugt werden. Dazu wird für jeden String $\mathbf{s}^{(k)}$ eine Wahrscheinlichkeit

$$p^{(k)} = \left[\frac{I^{(k)}}{N}\right]^{\eta} \qquad \text{mit} \qquad I^{(k)} = \sum_{i=1}^{N} s_i^{(k)} \tag{2.12}$$

definiert, die abhängig von der Anzahl der Einsen $I^{(k)}$ in einem String und einem Parameter η ist (s. Alg. 2.3.1, Schritt 6).

Der in [Banzhaf 1993b] vorgeschlagene Algorithmus sieht zusätzlich zum Reaktor-Algorithmus 2.1.2 eine wachsende Population und die Behandlung von Destruktoren und Exploiter vor:

Algorithmus 2.3.1 *Erweiterter Reaktor-Algorithmus nach [Banzhaf 1993b], S. 272:*

- *Initialisierung (Befüllung des Reaktors):*
  Erzeuge M_1 zufällige Binärstrings. Diese bilden die Anfangspopulation P.
- *Iteration (Betrieb des Reaktors):*
  1. *Wähle zufällig einen String* $\mathbf{s}$ *der Population und falte ihn zu einem Operator* P_s.
  2. *Wähle zufällig einen weiteren String* $\mathbf{s}'$ *und und wende den Operator* P_s *entsprechend Gleichung (2.11) an, um* $\mathbf{s}''$ *zu bilden.* $\mathbf{s}$ *und* $\mathbf{s}'$ *werden nicht aus der Population entfernt.*
  3. *Füge den neuen String* $\mathbf{s}''$ *der Population zu.*
  4. *Entferne mit der Wahrscheinlichkeit $p = M_1/M_2$ einen String aus der Population. (M_1: Aktuelle Populationsgröße, M_2: maximale Populationsgröße, $M_1 \leq M_2$).*
  5. *Ersetze jeden Destruktor der Population jeweils durch einen zufälligen String, so daß kein Destruktor in der Population verbleibt.*
  6. *Wähle zufällig einen String der Population und ersetze ihn mit der Wahrscheinlichkeit von (2.12) durch einen zufälligen String.*

Die Abbildung 2.6 zeigt den Konzentrationsverlauf während einer Simulation mit $N = 4$ und einer konstanten Populationsgröße von $M = M_1 = M_2 = 10^5$. Das entsprechende DGL-Modell für Algorithmus 2.3.1 ([Banzhaf 1993a] S. 5) hat die Form der katalytischen Netzwerkgleichung (2.5) mit zusätzlichen linearen Termen wegen der Destruktor- und Exploiterbehandlung in den Schritten 5 und 6.

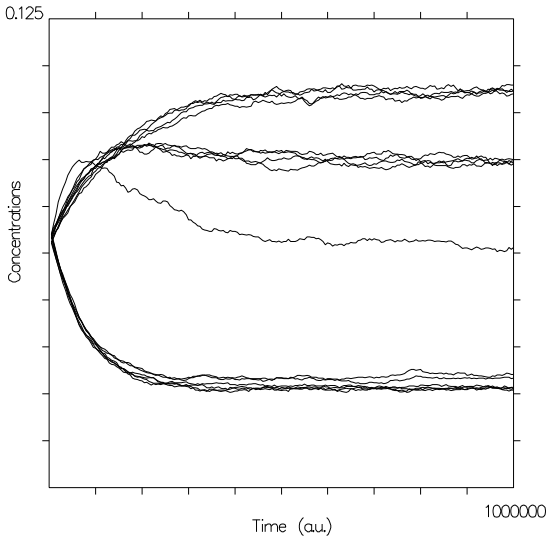


Abb. 2.6: *Konzentrationsverlauf während eines Laufs von Algorithmus 2.3.1 mit $N = 4$ und einer konstanten Populationsgröße von $M = M_1 = M_2 = 10^5$ unter Verwendung der Faltung 2.9 und der Geichung 2.11. (Quelle: [*Banzhaf 1993a*] S. 6).*

2.3.2 Algorithmische Chemie – ein funktionales Reaktionssystem

Walter Fontana hat zur Untersuchung von Organisationsformen ein Reaktionssystem entwickelt, in dem die Spezies Worte (also Zeichenketten) einer einfachen funktionalen Sprache sind, die von Church's λ-Kalkül [Church 1941] abgeleitet ist [Fontana 1990; Fontana 1991; Fontana 1994; Fontana und Buss 1994]. Das λ-Kalkül stellt die Semantik zur Verfügung, mit dem ein Wort als eine Funktion interpretiert werden kann. Man kann sich ein solches Wort wie einen einfachen LISP– bzw. Scheme-Ausdruck mit einem formalen Parameter vorstellen. Die Reaktion zwischen zwei Objekten $\mathbf{s}, \mathbf{s}'$ entspricht der einer Funktionskomposition. Dazu wird der Ausdruck ($\mathbf{s}$ '($\mathbf{s}'$)) evaluiert, indem jedes Auftreten des formalen Parameters in $\mathbf{s}$ durch den Operand $\mathbf{s}'$ ersetzt wird, und dann der resultierende Ausdruck ausgewertet wird. Dabei entsteht ein neuer Ausdruck, der im allgemeinen wiederum den formalen Parameter enthält (ausführlich: [Fontana 1991] S. 170 ff.). Da es sich um eine rekursive Sprache handelt, die die Mächtigkeit einer Turing-Maschine besitzt, kann eine derartige Auswertung beliebig oder gar unendlich lange dauern. Deshalb wird die Zeitspanne, die für eine Reaktion zur Verfügung steht, begrenzt. Zusätzlich gibt es eine Längenbeschränkung für Worte, um ein ungehemmtes Wachstum zu vermeiden.

Bei seinen Simulationen stellte Fontana fest, daß nach einiger Zeit sich die Vielfalt der Population erheblich reduzierte, und oft nur eine einzige selbstreplizierende Spezies übrig blieb. Die Ensemble-Strukturen die dabei am Ende dominierten nennt Fontana **Ebene-0-Organisationen** ([Fontana 1994] S. 86). Sie bestehen im allgemeinen aus nur wenigen eng gekoppelten Replikatoren (Abb. 2.7, links).

Verbietet man sämtliche Replikationen, so erhält man ein deutlich komplexeres Verhalten. Bei Fontana haben sich dabei Strukturen ausgebildet, die er **Ebene-1-Organisationen**

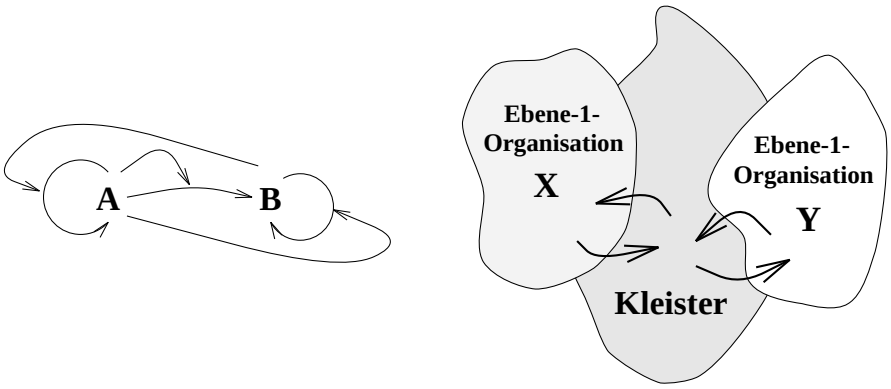


Abb. 2.7: *Links: Beispiel für eine Ebene-0-Organisation, die von zwei Spezies A und B gebildet wird. (Nach [*Fontana und Buss 1994*] Abb. 1a).*
*Rechts: Schema einer einfachen Ebene-2-Organisation, die zwei Ebene-1-Organisationen X und Y enthält. Die Ebene-1-Organisationen sind für sich selbsterhaltend. Sie erzeugen Kreuzungsprodukte (Kleister genannt) deren Wechselwirkungen teilweise wieder in X und Y münden, was zu einer Stabilisierung der Organisation führt. (Nach [*Fontana 1994*] Abb. 6).*

nennt. Sie werden durch erheblich mehr unterschiedliche Spezies getragen als die der Ebene 0 und sind äußerst stabil. Man kann eine ständige Produktion neuer Objekte beobachten. Die Menge aller dieser Objekte wird **Abschluß** der Organisation genannt und ist im Fall der Ebene 1 eine unendliche Teilmenge aller λ-Ausdrücke. Das bedeutet, daß die Ebene-1-Organisation immer nur durch einige ihrer Spezies in der Population vertreten ist. Die Organisation läßt sich mit Hilfe einer Grammatik und einer algebraischen Struktur unabhängig von λ charakterisieren (vgl. [Fontana 1994] S. 89). Eine von außen eingebrachte fremde Spezies wird nur selten aufgenommen. Im Falle der Aufnahme in die Organisation bleibt ihre ursprüngliche Struktur meist erhalten und wird nur "zwiebelförmig" um die neue Spezies und ihrer Reaktionsprodukte erweitert.

Eine **Ebene-2-Organisation** entsteht aus mehreren Ebene-1-Strukturen. Diese können allerdings nur stabil nebeneinander existieren, falls zwischen ihnen ein "Kleister" (engl. glue) entsteht, der aus Reaktionsprodukten von Spezies gebildet wird, die aus unterschiedlichen Ebene-1-Organisationen stammen (Abb. 2.7). Ebene-2-Organisationen entstehen spontan nur sehr selten ([Fontana 1994] S. 96). In den 32-Bit-Binärstringsystemen dieser Arbeit sind derartige Strukturen nicht spontan entstanden.

Kapitel 3

Operatorstrukturen für Binärstringsysteme

Dieses Kapitel gibt einen Überblick über Methoden der Informatik, mit deren Hilfe Reaktionen zwischen Binärstrings definiert werden können. Eine Reaktion $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ erfolgt dabei – außer im Abschnitt 3.1 – in zwei Schritten. Zunächst wird $\mathbf{s}$ zu einem Operator gefaltet. Dann wird dieser Operator auf $\mathbf{s}'$ angewendet, um $\mathbf{s}''$ zu bilden.

3.1 Ein zwei-dimensionales Feld als Reaktionsmatrix

Für kleine Stringlängen N kann die Reaktionsmatrix als zwei-dimensionales Feld (engl. look-up-table) implementiert werden. Der dazu benötigte Speicherplatz beträgt ca. $N2^{2N}$ Bits. Die folgende Tabelle gibt den Speicherbedarf exemplarisch für einige Stringlängen an:

Stringlänge N N	Stringanzahl n $n = 2^N$	Speicheraufwand m $m = N2^{2N}$ Bits	Speicheraufwand in CD-ROMs 1 CD-ROM = 4.8×10^9 Bits
1 Bit	2	4 Bits	1 CD-ROM
4 Bits	16	1 024 Bits	1 CD-ROM
9 Bits	512	2 359 296 Bits	1 CD-ROM
16 Bits	65 536	68 719 476 736 Bits	15 CD-ROMs
32 Bits	4 294 967 296	5.902×10^{20} Bits	ca. 122 978 300 000 CD-ROMs

Es ist also unmöglich, die Interaktionsprodukte zwischen langen Strings explizit als zwei-dimensionales Feld anzugeben. Vielmehr muß für lange Strings die Definition eines Reaktionssystems implizit erfolgen.

3.2 Boolesche Netzwerke

Die konsequenteste Art und Weise, das Medium Digitalelektronik zu respektieren, besteht darin, einen Binärstring zu einem booleschen Netzwerk zu falten. Für eine Reaktion $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ wird zunächst $\mathbf{s}$ zu einer digitalen Schaltung transformiert, die über n_e

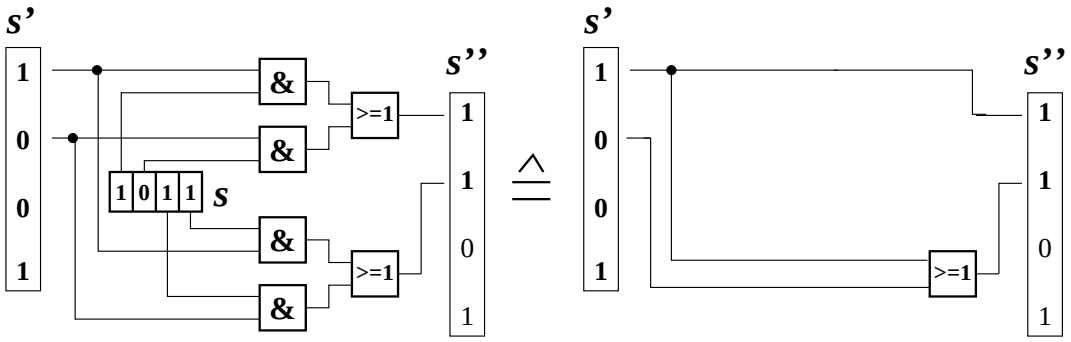


Abb. 3.1: *Beispiel, wie die Faltung eines Binärstrings* $\mathbf{s} = (1, 0, 1, 1)$ *zu einer kombinatorischen booleschen Schaltung erfolgen kann. Dazu wird ein internes Register der Schaltung mit dem Wert von* $\mathbf{s}$ *geladen, wodurch die Funktionalität der Schaltung festgelegt wird. In diesem speziellen Fall wird die gleiche Reaktion realisiert wie durch* Banzhaf*'s Operator* P_s *(vgl. Abb. 2.5).*

Eingangsleitungen und n_a Ausgangsleitungen verfügt. Im einfachen Fall einer **kombinatorischen Schaltung** wird eine boolesche Funktion $f : \mathbb{B}^{n_e} \rightarrow \mathbb{B}^{n_a}$ realisiert, die analog zu Banzahf's Matrix P_s verwendet werden kann, um schrittweise den String $\mathbf{s}'$ auf $\mathbf{s}''$ abzubilden.

Auf dem Gebiet der genetischen Algorithmen (GAs) finden sich eine Fülle von Techniken, Binärstrings auf Netzwerkstrukturen abzubilden (z.B. [Miller, Todd, und Hedge 1989] oder [Höffgen, Siemon, und Ultsch 1991]).

Ein weiterer besonders interessanter Ideenlieferant sind die programmierbaren Logikbausteine (PLDs) der Elektronik. Bei diesen Bausteinen kann die interne Verdrahtung "per Software" verändert werden. Die Verschaltung hängt dabei von internen Zustandsregistern ab, die bei einigen Typen sogar beliebig oft neu geladen werden können. Abbildung 3.1 zeigt ein einfaches Beispiel für Binärstrings der Länge 4.

3.3 Zelluläre Automaten

Wie das Paradigma des zellulären Automaten (ZA) für die Definition einer Reaktion verwendet werden kann, wird beispielhaft anhand eines ein-dimensionalen Automaten mit 32 Zellen gezeigt:

Der Automat besteht hier aus 32 nebeneinander liegenden **Zellen**, deren Zustände zu einem Zeitpunkt t durch $z_1(t), z_2(t), \ldots, z_{32}(t)$ mit $z_i \in \mathbb{B}, \mathrm{i} = 1, \ldots, 32$ gegeben sind, und aus einer **Übergangsfunktion** $\mathrm{F} : \mathbb{B}^5 \rightarrow \mathbb{B}$. Der Zustand einer Zelle zum nächsten Zeitpunkt $t + 1$ wird durch ihren Zustand und den Zuständen der Nachbarzellen (hier 4) zum Zeitpunkt t festgelegt:

$$z_i(t+1) = \mathrm{F}(z_{i-2}(t), z_{i-1}(t), z_i(t), z_{i+1}(t), z_{i+2}(t)) \qquad i = 1, \ldots, 32 \tag{3.1}$$

Die Anordnung der Zellen ist zyklisch, so daß zum Beispiel z_{-1} äquivalent zu z_{31} ist.

Die Reaktion $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ zwischen zwei Binärstrings der Länge $N = 32$ wird folgendermaßen durchgeführt (Abb. 3.2):

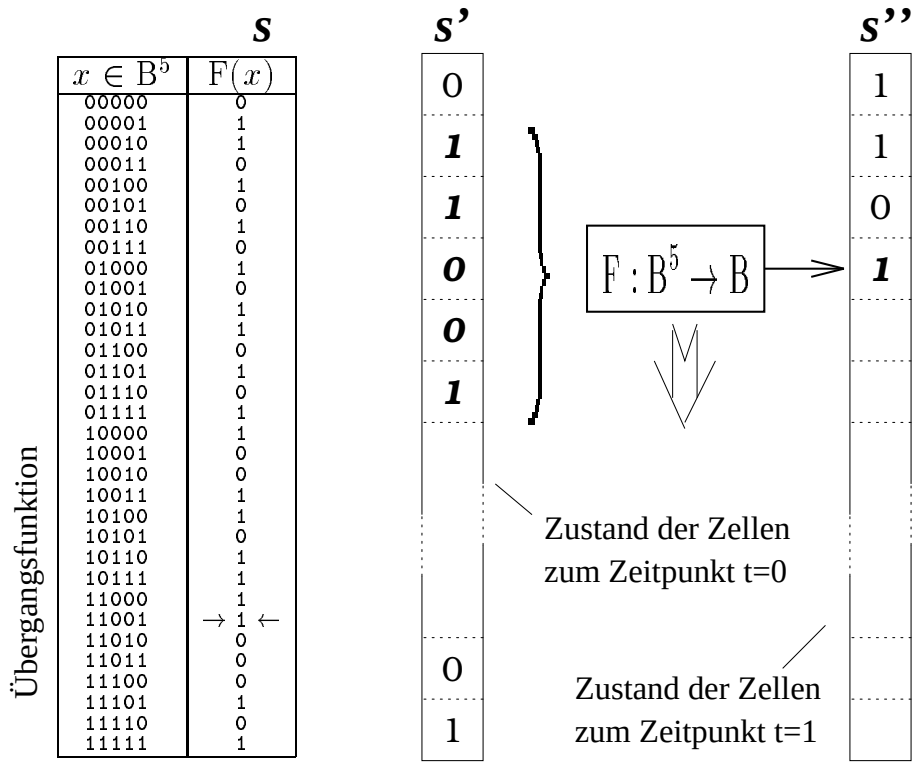


Abb. 3.2: *Reaktion* $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ *zwischen Binärstrings der Länge 32 mit Hilfe eines zellulären Automaten, dessen Übergangsfunktion aus* $\mathbf{s}$ *gebildet wird. Die Zellen des Automaten werden zum Zeitpunkt 0 mit dem Operandenstring* $\mathbf{s}'$ *geladen. Nach einem Zeitschritt enthalten sie den Resultatstring* $\mathbf{s}''$.

1. Falte $\mathbf{s}$ zu der Übergangsfunktion $\mathrm{F}_s : \mathbb{B}^5 \to \mathbb{B}$, indem einfach $\mathbf{s}$ als Lookup-Tabelle verwendet wird. Das heißt, daß $\mathrm{F}_s(x)$ das x-te Bit von $\mathbf{s}$ liefert. Formal: $\mathrm{F}_s(x) = s_x$, $s_x \in \mathbb{B}$ mit $x \in \mathbb{B}^5$ und $\mathbf{s} = (s_{0000}, s_{0001}, s_{0010}, \ldots, s_{1111})$.

2. Lade die Zellen z_i des zellulären Automaten zum Zeitpunkt $t = 0$ mit dem Operanden $\mathbf{s}'$ entsprechend $z_i(0) :\overline{=}\ s'_i = 1, \ldots, 32$.

3. Führe einen Zeitschritt aus (nach Gl. 3.1 mit $\mathrm{F} = \mathrm{F}_s$). Das Reaktionsprodukt entspricht dem neuen Zustand des ZA: $s''_i = z_i(1)$.

Einen solchen ZA nennt man **homogen**, da für jede Zelle die gleiche Übergangsfunktion gilt.

Bei allen ad hoc Experimenten mit dem Reaktor-Algorithmus, bei dem obiger homogener ZA für die Erzeugung der Reaktionsprodukte verwendet wurde, dominierte nach einiger Zeit entweder der String 000 ... 000 oder 111 ... 111. Verbietet man diese beiden Destruktoren, so bleiben Strings übrig, die sich nur in einer Bitposition von einem der beiden Destruktoren unterscheiden (z.B. 000100 ... 00).

Führt man eine Inhomogenität ein, indem einfach eine ausgezeichnete Zelle k des Automaten vor der Ausführung des Zeitschritts invertiert wird, so erhält man ein chaotisches

```
0000 1000 1111              a := READ
0001 1000 1111              b := READ

1100 0000                   IF a THEN
1110 0001                        WRITE b
1110                        END_IF

0010                        DO
0001 1000 0001 1001 1111         b := a EXOR READ
0100                        WHILE_NOT_EOF
```

Abb. 3.3: *Konzept der Faltung eines Binärstrings der Länge 72 zu einem imperativen Programm.* `a` *und* `b` *sind boolesche Variablen. Der Befehl* `READ` *ließt das nächste Bit vom Eingabestrom. Die Schleife* `DO ... WHILE_NOT_EOF` *wird so lange durchlaufen, bis das Ende des Eingabestroms erreicht wird.*

Verhalten. Kein String macht sich durch einen signifikanten Konzentrationsanstieg bemerkbar. Selbst bei einer Populationsgröße von 10^5 ist kaum ein String mehr als einmal vorhanden.

Da bei allen ad hoc Experimenten mit der ZA-Reaktion kein derartig komplexes Verhalten wie bei der Automaten-Reaktion (vgl. Abschnitt 3.6) aufgetreten ist, wird sie hier nicht weiter systematisch untersucht.

3.4 Imperative Programme

Ein **imperatives Programm** besteht aus einer Folge von Anweisungen. Die Ein-Ausgabe erfolgt über Seiteneffekte – hier durch die Befehle `READ` und `WRITE`. Ferner gibt es einen eindeutigen Kontrollfluß.

Ein Binärstring $\mathbf{s}$ kann in ein Programm übersetzt werden, indem – wie bei einem Assembler-Kode – Stringabschnitte als Nummern für Befehle der Sprache interpretiert werden. Die Abbildung 3.4 zeigt ein Beispiel, bei dem jeweils Gruppen von 4 Bits einen Befehl kodieren. Für die Reaktion $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ wird das Programm gestartet, wobei $\mathbf{s}'$ der Eingabestrom und $\mathbf{s}''$ der Ausgabestrom ist. Das heißt, daß mit jedem `READ`-Befehl jeweils das nächte Bit von $\mathbf{s}'$ gelesen wird, und jeder `WRITE`-Befehl den Resultatstring $\mathbf{s}''$ um ein Bit verlängert.

3.5 Die Typogenetik – eine abstrakte Maschine

Die Typogenetik wurde 1979 von Douglas R. Hofstadter in seinem Buch "Gödel, Escher, Bach" eingeführt (deutsche Ausgabe: [Hofstadter 1985] S. 540 ff.) und später von H. C. Morris intensiv untersucht [Morris 1988]. Die Typogenetik ist ein formales System, das die "molekulare Logik des Lebens" einfangen soll.

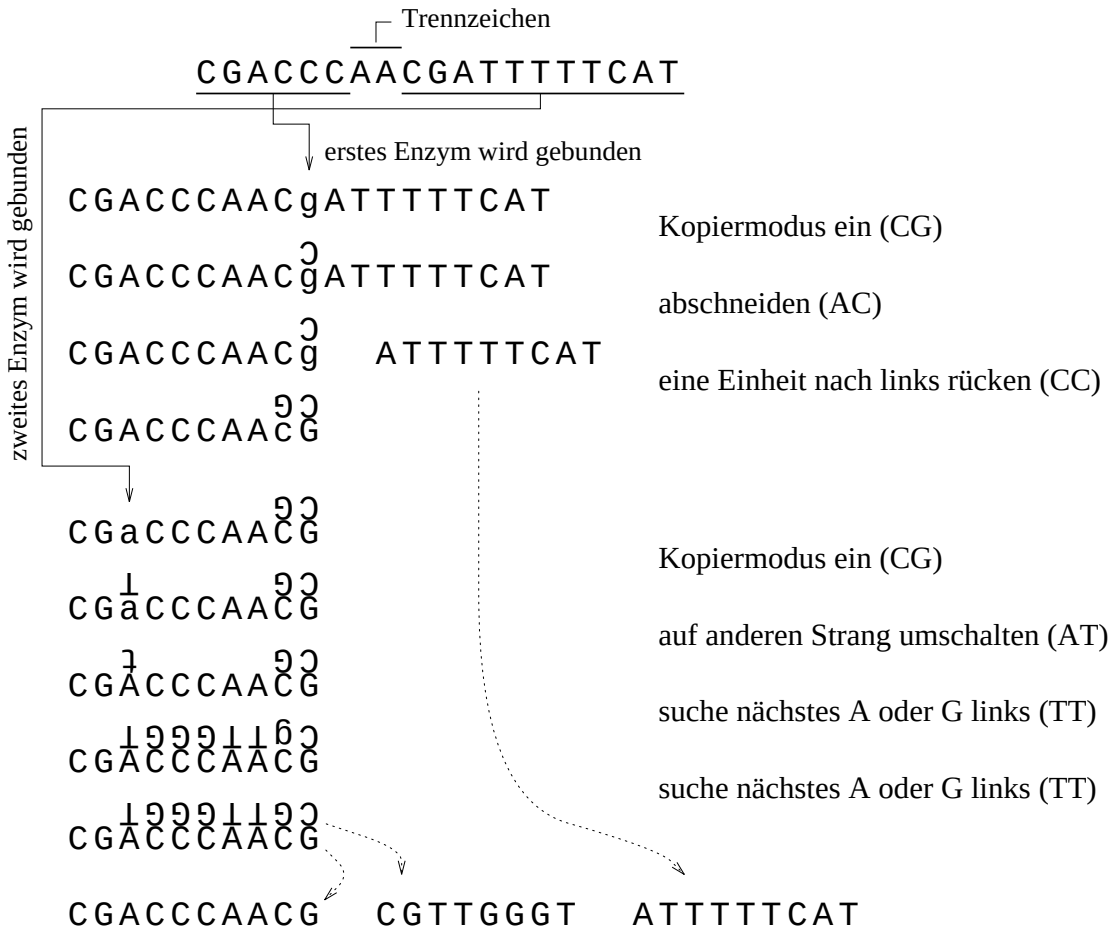


Abb. 3.4: *Die Typogentik in Aktion. Es wird gezeigt, wie die zwei Enzyme, die der Strang* `CGA....AT` *codiert, auf einer Instanz desselbigen arbeiten und dabei drei Reaktionsprodukte erzeugen. Die aktuelle Bindungsposition eines Enzyms an einen Strang wird durch einen Kleinbuchstaben symbolisiert. (Beispiel entnommen aus [*Morris 1989*]).*

Das System besteht aus der typographischen Manipulation von Zeichenketten, die aus den Buchstaben `A`, `C`, `G` und `T` gebildet werden – analog zu den Basen, aus denen die Nukleinsäuren bestehen. Die Manipulation erfolgt durch Operationen, wie zum Beispiel "zerschneiden", "einfügen" eines bestimmten Zeichens oder "löschen" eines Zeichens. Mehrere Operationen werden zu einem Paket zusammengefaßt, das sich auf einer Zeichenkette – wie eine Turing-Maschine auf ihrem Band – hin und her bewegt und dabei die Manipulationen ausführt (Abb. 3.4). Analog zur Sprache der Genetik, wird ein Operationenpaket **Enzym** und eine Zeichenkette **Strang** genannt.

Ein Strang kodiert eine Reihe von Enzymen, deren "genetische Bauanleitungen" durch die zwei Buchstaben `AA` voneinander getrennt werden. Zur Übersetzung eines Teilstücks in eine Operationenfolge werden jeweils zwei Zeichen zusammengefaßt. Zum Beispiel bedeutet `AC` "abschneiden", was bewirkt, daß rechts von der aktuellen Bindungsposition des Enzyms der Strang getrennt wird (Abb. 3.4).

Die **Bindungspräferenz** eines Enzyms hängt in nicht-trivialer Weise von all seinen Bausteinen ab (vgl. [HOFSTADTER 1985] S. 546). Für das erste Enzym des Strangs in der Abbildung 3.4 ist dies das `G`. Enthält der zu bearbeitende Strang mehrere dieser Zeichen, so wird eines zufällig für die Startposition des Enzyms ausgewählt.

Eine interessante Eigenschaft ist die Möglichkeit zur Ausbildung eines Doppelstrangs. Dazu gibt es zwei Operationen, die einen **Kopiermodus** ein- bzw. ausschalten. Ist der Kopiermodus aktiv, so werden bei jeder Bewegung des Enzyms die dabei "berührten" Zeichen kopiert. Allerdings erfolgt keine identische Kopie, sondern eine Komplementbildung. Dabei wird `A` auf `T` und `G` auf `C` abgebildet und umgekehrt. Durch den Befehl "umschalten" (`AT`) kann die Bindung eines Enzyms sogar während der Bearbeitung auf den so erzeugten Parallelstrang wechseln (Abb. 3.4).

Von den bisher vorgestellten Techniken ist die Typogenetik äußerst interessant. Allerdings widerspricht ihre hohe Komplexität dem Einfachheits-Prinzip. Problematisch ist insbesondere: 1.) Die Stranglänge ist variabel. 2.) Bei einer Reaktion können mehrere Produkte entstehen. 3.) Aufgrund der zufälligen Bindungsposition eines Enzyms ist die Reaktion nicht-deterministisch.

3.6 Die Automaten-Reaktion

In diesem Abschnitt wird eine eindeutige Reaktion zwischen Binärstrings von konstanter Länge $N = 32$ definiert, die für alle Experimente im Kapitel 4 verwendet wird. Es handelt sich dabei um eine vereinfachte Typogenetik. Die Vereinfachungen erfolgen im Hinblick auf die Vorgabe "Respektiere das Medium", so daß sich die Reaktion leicht als Hardware realisieren läßt.

Um die Reaktion $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ durchzuführen, wird zunächst $\mathbf{s}$ zu einem Automaten gefaltet. Dann wird der Automat mit $\mathbf{s}'$ als Eingabe gestartet. Er ist so konstruiert, daß er in jedem Fall hält.

Der Automat verfügt über zwei 32-Bit-Register (Abb. 3.5). Das **Operatorregister** wird zu Beginn mit $\mathbf{s}$ und das Operandenregister mit $\mathbf{s}'$ geladen. Diese Register haben die gleiche Funktion wie das Band einer Turing-Maschine. Sie sind allerdings endlich und "nur" 32 Bit lang.

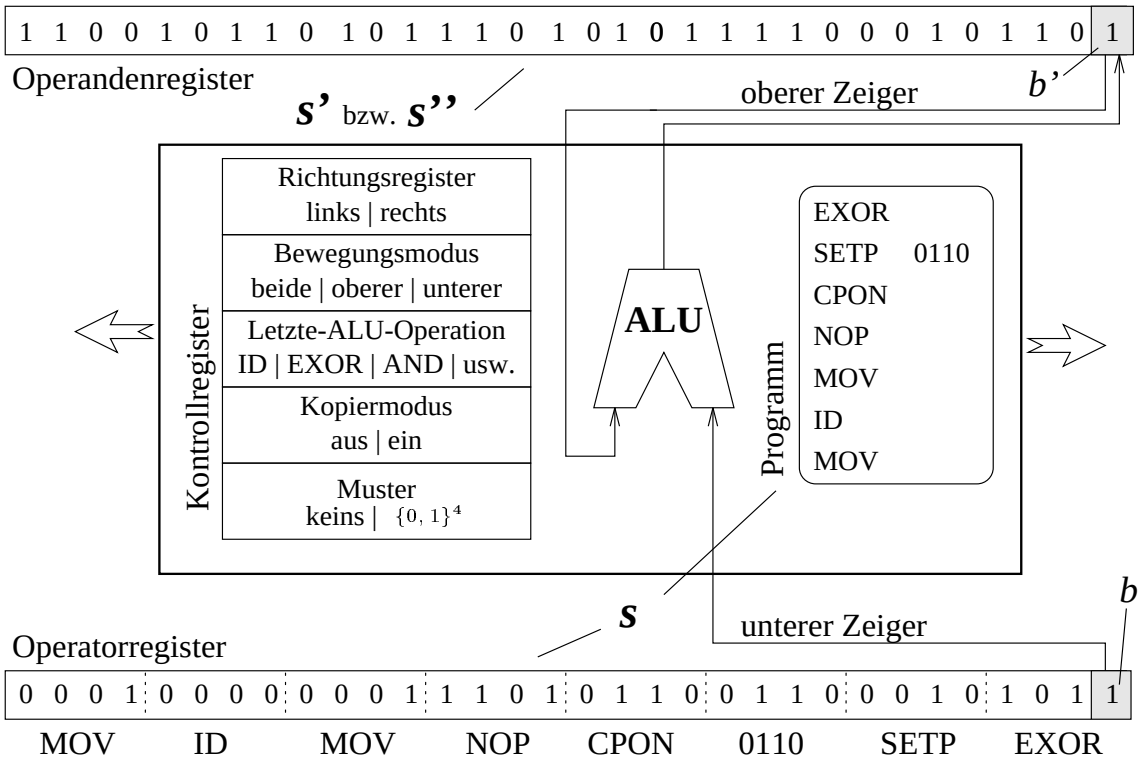


Abb. 3.5: *Schema des Automaten, der durch die Faltung von* s *entsteht. Der Automat arbeitet ähnlich wie eine Turing-Maschine oder die Typogenetik, mit dem hauptsächlichen Unterschied, daß die Band- bzw. Stranglänge nicht beliebig sondern konstant gleich 32 Bit ist. Es wird die Reaktion* $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ *realisiert.* s *wird zu Beginn in das Operatorregister geschrieben und legt das Programm fest. Das Operandenregister wird mit* $\mathbf{s}'$ *initialisiert. Nach Abarbeitung des Programms steht das Resultat* $\mathbf{s}''$ *ebenfalls im Operandenregister.*

Für jedes Register gibt es einen **Zeiger**, der eine Bit-Position des jeweiligen Registers markiert. Der **obere Zeiger** zeigt auf ein Bit des Operandenregisters, der **untere Zeiger** auf ein Bit des Operatorregisters. Die **ALU** kann diese beiden Bits miteinander verknüpfen. Das Resultat – ebenfalls 1 Bit – wird in das Operandenregister geschrieben (Abb. 3.5).

Der Operatorstring **s** legt das interne **Programm** fest, indem jeweils 4 Bits zu einem Befehl zusammengefaßt werden. Die Abbildung 3.6 zeigt zwei Kodes, die dafür eingesetzt werden. Wird der Automat gestartet, so werden die Befehle nacheinander, beginnend mit dem ersten Befehl, abgearbeitet. Der Automat hält nach Beendigung des letzten Befehls. Bis auf den Befehl STOP gibt es keine Operationen, die den Kontrollfluß steuern, wie zum Beispiel Sprung- oder Schleifenoperationen.

3.6.1 Die Kontrollregister

Neben den beiden 32-Bit-Registern verfügt der Automat über 5 kleinere **Kontrollregister**. Die Kontrollregister werden mit dem ersten Wert des angegebenen Wertebereichs initialisiert.

Kode I	
0000	ID
0001	MOV
0010	SETP
0011	TMM
0100	TDIR
0101	UNSETP
0110	CPON
0111	CPOFF
1000	STOP
1001	OR
1010	NOT
1011	EXOR
1100	EXOR
1101	NOP
1110	ID
1111	AND

Kode II	
0000	ID
0001	MOV
0010	SETP
0011	TMM
0100	TDIR
0101	UNSETP
0110	CPON
0111	CPOFF
1000	STOP
1001	OR
1010	EQ $\Leftarrow$
1011	EXOR
1100	EXOR
1101	NOP
1110	ID
1111	AND

Kode III	
0000	ID
0001	MOV
0010	SETP
0011	TMM
0100	TDIR
0101	UNSETP
0110	CPON
0111	CPOFF
1000	STOP
1001	OR
1010	ID $\Leftarrow$
1011	EXOR
1100	EXOR
1101	NOP
1110	ID
1111	AND

Abb. 3.6: *Verwendete Kodes für die Übersetzung eines Binärstrings in ein Programm. Die Kodes II und III enthalten gegenüber dem Kode I keine NOT-Operation.*

Richtungsregister Wertebereich: links, rechts
Das Richtungsregister legt die Bewegungsrichtung der Zeiger bei einem Bewegungsbefehl fest. Der Befehl TDIR kehrt die Richtung um.

Bewegungsmodus Wertebereich: beide, unterer, oberer
Dieses Register bestimmt, welcher Zeiger bei einem Bewegungsbefehl bewegt wird. Der Befehl TMM schaltet den Bewegungsmodus um.

Letzte-ALU-Operation Wertebereich: ID, NOT, AND, OR, EXOR, EQ
Dieses Register hält die letzte ALU Operation fest. Ist der Kopiermodus eingeschaltet, so wird die hier gespeicherte Operation bei jeder Bewegung durchgeführt.

Kopiermodus Wertebereich: aus, ein
Ist der Kopiermodus eingeschaltet, so erfolgt bei jedem Bewegungsschritt der Zeiger eine ALU-Operation entsprechend dem Zustand des Registers "Letzte-ALU-Operation".

Muster Wertebereich: keins, $\{0,1\}^4$
Durch den Befehl SETP wird in diesem Register ein Muster gespeichert. Mit dem Befehl UNSETP kann das Muster gelöscht werden. Ist ein Muster gesetzt, so werden bei jedem Bewegungsbefehl (MOV) die Zeiger solange einen Schritt weiterbewegt, bis sie das Muster erreichen; maximal 32 Schritte. Werden beide oder nur der untere Zeiger bewegt, so wird das Muster im Operator-

register gesucht. Wird nur der obere Zeiger bewegt, so erfolgt die Suche im Operandenregister.

3.6.2 Arithmetische und logische Operationen

Bei einer arithmetischen oder logischen Operation wird das Bit b, auf das der untere Zeiger deutet, mit dem Bit b', auf das der obere Zeiger weist, verknüpft. Das Ergebnis wird an die Stelle von b' geschrieben. Danach werden die Zeiger entsprechend dem Richtungsregister eine Bitposition weiter bewegt.

ID	Identität: $b' := b$.
NOT	Negation: $b' := \neg b$.
AND	UND-Verknüpfung: $b' := b \wedge b'$.
OR	ODER-Verknüpfung: $b' := b \vee b'$.
EXOR	Exklusiv-ODER: $b' := b \neq b'$.
EQ	Gleichheit: $b' := b = b'$.

3.6.3 Bewegungs- und Kontrolloperationen

MOV — Bewegung der Zeiger entsprechend dem Richtungsregister, dem Bewegungsmodus und dem gesetzten Muster.
Ist kein Muster durch den Befehl SETP gesetzt, so erfolgt eine Bewegung um einen Schritt in die Richtung, die durch das Richtungsregister angegeben wird. Ist ein Muster (4 Bits) gesetzt, so werden die Zeiger solange einen Schritt weiter bewegt, bis das Muster gefunden wird; maximal 32 Schritte. Wird laut Bewegungsmodus nur der obere Zeiger bewegt, so erfolgt die Suche im Operandenregister. Ansonsten wird das Muster im Operatorregister gesucht.
Ist der Kopiermodus eingeschaltet (durch CPON), so erfolgt bei jedem Bewegungsschritt eine logische Operation entsprechend dem Register "Letzte-ALU-Operation".

CPON — Schaltet den Kopiermodus ein: Kopiermodus := ein.

CPOFF — Schaltet den Kopiermodus aus: Kopiermodus := aus.

SETP **p** — Legt das Muster $\mathbf{p} \in \{0,1\}^4$ für die Bewegung fest: Muster := **p**.
Dies ist der einzige Befehl, der über ein Argument verfügt.

UNSETP — Löscht das Muster: Muster := keins.

TDIR — Kehrt die Bewegungsrichtung um:

$$\text{Richtungsregister} := \begin{cases} \text{rechts} & \text{falls Richtungsregister} = \text{links}, \\ \text{links} & \text{falls Richtungsregister} = \text{rechts} \end{cases}$$

TMM Schaltet den Bewegungsmodus um:

$$\text{Bewegungsmodus} := \begin{cases} \text{beide} & \text{falls Bewegungsmodus} = \text{unterer}, \\ \text{oberer} & \text{falls Bewegungsmodus} = \text{beide}, \\ \text{unterer} & \text{falls Bewegungsmodus} = \text{oberer} \end{cases}$$

NOP Bewirkt nichts.

STOP Stopt den Automaten vorzeitig.

3.6.4 Ein Beispiel der Automaten-Reaktion

Das folgende Beispiel zeigt, wie mit Hilfe des soeben definierten Automaten die Reaktion $\mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}''$ mit

$$\mathbf{s} = 0001\,0000\,0001\,1101\,0110\,0110\,0010\,1011$$
$$\mathbf{s}' = 1100\,1011\,0101\,1101\,0101\,1110\,0010\,1101$$

durchgeführt wird. Der Operatorstring $\mathbf{s}$ legt das Programm des Automaten nach Kode I fest:

Kode	Operation
1011	EXOR
0010 0110	SETP 0110
0110	CPON
1101	NOP
0001	MOV
0000	ID
0001	MOV

Der Automat wird folgendermaßen initialisiert:

$$\begin{aligned} \text{Operatorregister} &:= \mathbf{s} \\ \text{Operandenregister} &:= \mathbf{s}' \\ \text{Richtungsregister} &:= \text{links} \\ \text{Bewegungsmodus} &:= \text{beide} \\ \text{Letzte-ALU-Operation} &:= \text{ID} \\ \text{Kopiermodus} &:= \text{aus} \\ \text{Muster} &:= \text{keins} \end{aligned}$$

Der untere und der obere Zeiger werden auf die erste Bit-Position von rechts der entsprechenden Register gesetzt (Abb. 3.5). Die Abbildung 3.7 zeigt die Arbeitsschritte des Automaten. Das Resultat lautet:

$$\mathbf{s}'' = 1100\,1011\,0101\,1101\,0101\,0110\,0000\,0110$$

Operation	Register	Zustand nach der Operation
Initialisierung	Operandenregister	1100 1011 0101 1101 0101 1110 0010 110*1*
	Operatorregister	0001 0000 0001 1101 0110 0110 0010 101*1*
EXOR	Operandenregister	1100 1011 0101 1101 0101 1110 0010 11*0***0**
	Operatorregister	0001 0000 0001 1101 0110 0110 0010 10*1***1**
	Letzte-ALU-Operation	EXOR
SETP 0110	Operandenregister	1100 1011 0101 1101 0101 1110 0010 11*0*0
	Operatorregister	0001 0000 0001 1101 0110 0110 0010 10*1*1
	Muster	0110
NOP	Operandenregister	1100 1011 0101 1101 0101 1110 0010 11*0*0
	Operatorregister	0001 0000 0001 1101 0110 0110 0010 10*1*1
CPON	Operandenregister	1100 1011 0101 1101 0101 1110 0010 11*0*0
	Operatorregister	0001 0000 0001 1101 0110 0110 0010 10*1*1
	Kopiermodus	ein
MOV	Operandenregister	1100 1011 0101 1101 0101 111*0* **0000 011**0
	Operatorregister	0001 0000 0001 1101 0110 011*0* **0010 101**1
ID	Operandenregister	1100 1011 0101 1101 0101 11*1***0** 0000 0110
	Operatorregister	0001 0000 0001 1101 0110 01*1***0** 0010 1011
	Letzte-ALU-Operation	ID
MOV	Operandenregister	1100 1011 0101 1101 010*1* **011**0 0000 0110
	Operatorregister	0001 0000 0001 1101 011*0* **011**0 0010 1011

Abb. 3.7: *Arbeitsweise der Automaten-Reaktion für das Beispiel im Abschnitt 3.6.4. Die Position der Zeiger ist durch kursive Schrift angedeutet. Die Stellen, die in der letzten Operation involviert sind, werden fett gedruckt. Das Programm besteht hier aus 7 Operationen, da der Befehl "SETP 0110" durch 8 bits Kodiert ist.*

3.6.5 Eigenschaften zufällig erzeugter Spezies

Zieht man zufällig Spezies aus $\mathbb{B}^{32}$, so ist im Mittel jede zweite (s.u.) ein **Selbstreplikator**, also eine Spezies $\mathbf{s}$ für die gilt: $\mathbf{s} + \mathbf{s} \Longrightarrow \mathbf{s}$.

Spezielle Selbstreplikatoren sind die "faulen" und die "fleißigen" Replikatoren.

Definition 3.6.1 *Eine Spezies* $\mathbf{s}$ *heißt* **fauler Replikator** $:\Leftrightarrow \quad \forall \mathbf{s}' \in S : \mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}'$.

Ein fauler Replikator tut praktisch nichts, so daß das Operandenregister nicht verändert wird. Dadurch repliziert er nicht nur sich selbst, sondern auch jede andere Spezies. Ein Beispiel ist ein String, der mit der Binärsequenz für den STOP-Befehl beginnt. Fast jede vierte zufällig gezogene Spezies ist ein fauler Replikator.

Definition 3.6.2 *Eine Spezies* $\mathbf{s}$ *heißt* **fleißiger Replikator** $:\Leftrightarrow \quad \forall \mathbf{s}' \in S : \mathbf{s} + \mathbf{s}' \Longrightarrow \mathbf{s}$.

Fleißige Replikatoren entstehen sehr selten – mit einer Wahrscheinlichkeit von ca. 0.004 %. Ein fleißiger Replikator kopiert den Inhalt des Operatorregisters in das Operandenregister, und repliziert sich somit unabhängig vom Operanden $\mathbf{s}'$. Fleißige Replikatoren setzen sich

– falls sie auftauchen, und die Replikation erlaubt ist – gegen alle anderen Strings durch. Sie sind die Destruktoren der Automaten-Reaktion.

Der Abschätzung obiger Wahrscheinlichkeiten, liegt (je Kode) ein Sample von $5 * 10^5$ zufällig erzeugten Binärstrings zugrunde. Die Eigenschaften jedes einzelnen Strings wurden anhand 20 zufällig erzeugter Referenzspezies getestet. Ein String wurde als fauler Replikator gezählt, falls er als Operator alle 20 Referenzspezies replizieren konnte. Er galt als fleißiger Replikator, falls er als Operator mit jeder der 20 Referenzspezies zu sich selbst reagiert hat. Dabei konnte kein signifikanter Unterschied zwischen den drei Kodes festgestellt werden.

Kapitel 4

Simulationen und Beobachtungen

In diesem Kapitel werden drei Versuchsreihen vorgestellt, die mit der Automaten-Reaktion durchgeführt worden sind. Eine **Versuchsreihe** besteht aus mehreren Läufen des Reaktor-Algorithmus, die sich ausschließlich in der Wahl der Populationsgrößen unterscheiden. In der Regel sind zu allen vorgestellten Populationsgrößen mehrere Läufe mit unterschiedlichen Seeds für den Zufallszahlengenerator erfolgt. Übersicht der Versuchsreihen:

Versuchsreihe	Abschnitt	Kode	Besonderheit
A	4.1	I	Reaktor-Algorithmus mit Automaten-Reaktion
B	4.2	I	wie A, aber *ohne* Replikation
C	4.3	II	wie B, aber *ohne* NOT-Operation

In den Versuchsreihen B und C wird die Replikation verboten, indem jede Reaktion als elastisch definiert wird, bei der das Produkt entweder gleich dem Operator- oder dem Operandenstring ist. Das hat unter anderem den Vorteil, daß eine neue Meßgröße hinzukommt: die Produktivität.

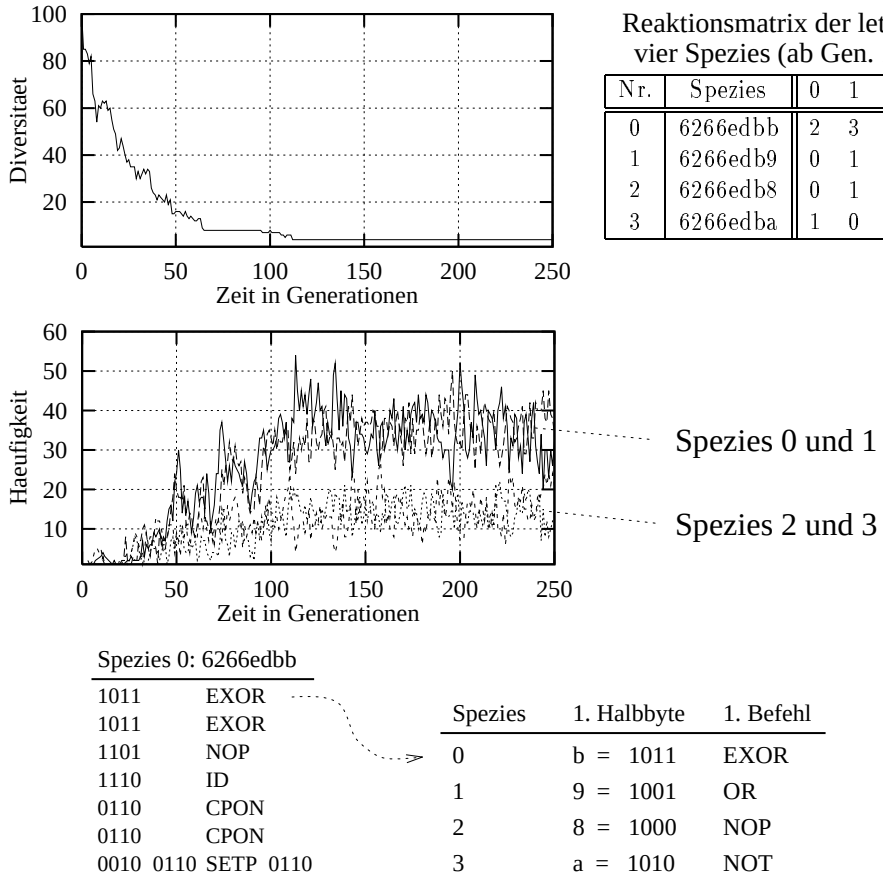
Die **Produktivität** ist der Anteil der Kollisionen während einer Generation, bei denen ein Reaktionsprodukt entstanden und in die Population eingefügt worden ist (vgl. Abschnitt 2.1). Für die Versuchsreihe A, in der die Replikation zugelassen wird, ist die Produktionsrate konstant gleich eins.

Unter der **Diversität** wird hier die absolute Anzahl unterschiedlicher Spezies in der Population verstanden. Die Diversität ist eine brauchbare Meßgröße für alle drei Versuchsreihen.

Die 32 Bit großen Binärstrings, werden im folgenden oft hexadezimal notiert. Dabei entspricht jedem Befehl ein Zeichen der Hexdezimaldarstellung. In einer Reaktionsmatrix erhält jeder String zusätzlich eine laufende Nummer, wobei gilt, je kleiner die Nummer, desto höher die Konzentration zum Zeitpunkt, an dem die Matrix aufgestellt worden ist. Die häufigste Spezies erhält immer die Nummer 0.

4.1 Versuchsreihe A – Simulationen *mit* Replikation

Diese Versuchsreihe enthält Läufe des Reaktor-Algorithmus 2.1.2, wobei die Reaktion durch die Automaten-Reaktion unter der Verwendnung von Kode I realisiert wird. Für



Reaktionsmatrix der letzten vier Spezies (ab Gen. 122)

Nr.	Spezies	0	1	2	3
0	6266edbb	2	3	0	1
1	6266edb9	0	1	1	0
2	6266edb8	0	1	2	3
3	6266edba	1	0	0	1

Spezies 0: 6266edbb

```
1011        EXOR
1011        EXOR
1101        NOP
1110        ID
0110        CPON
0110        CPON
0010 0110 SETP 0110
```

Spezies	1. Halbbyte	1. Befehl
0	b = 1011	EXOR
1	9 = 1001	OR
2	8 = 1000	NOP
3	a = 1010	NOT

Abb. 4.1: *Typischer Run der Versuchsreihe A mit einer kleinen Populationsgröße. $M = 100$, Kode I, Replikation erlaubt, Run 1. Erläuterung im Abschnitt 4.1.1.*

die Populationsgrößen $M = 100$ und $M = 1000$ sind jeweils 10 Läufe, für $M = 10000$ 5 Läufe mit unterschiedlichen Seeds durchgeführt worden. Die simulierte Zeit beträgt für jeden Lauf 1000 Generationen, also $1000 * M$ Iterationen bzw. Kollisionen.

4.1.1 Verhalten einer kleinen Population ($M = 100$)

In der folgenden Tabelle ist die Diversität der letzten Generation für alle 10 Runs der Populationsgröße $M = 100$ dargestellt:

Run	1	2	3	4	5	6	7	8	9	10
Diversität der Gen. 999	4	2	10	1	2	1	2	1	3	1

Der Run 1 ist, als im qualitativen Verhalten typischer Vertreter, in der Abbildung 4.1 dargestellt. Man erkennt, daß die Diversität sofort stark absinkt. Schnell ist ihr Minimum – in diesem Fall 4 Spezies – erreicht. Laut Reaktionsmatrix führt jede Kollision zwischen diesen Spezies zu einem String, der sich schon in der Population befindet. Die Organisation ist also abgeschlossen und nicht in der Lage neue Spezies zu erzeugen. Der Häufigkeitsverlauf der vier Spezies deutet darauf hin, daß das System in einen asymptotisch stabilen Fixpunkt

gelaufen ist. Die starken Schwankungen sind auf die nicht-deterministischen Elemente des Reakor-Algorithmus zurückzuführen, die sich besonders bei derartig kleinen Populationen bemerkbar machen (vgl. Abb. 2.1).

Ein weiteres Indiz für die Stabilität der Organisation ist die enge Verknüpfung der Spezies in der Reaktionsmatrix. Im Prinzip enthält die Reaktionsmatrix eine Reihe eng gekoppelter Reaktionszyklen. Zum Beispiel einen Zyklus der Länge 3: $0 + 1 \Longrightarrow 3$, $3 + 3 \Longrightarrow 1$, $1 + 3 \Longrightarrow 0$; oder einen Hyperzyklus der Länge 2: $0 + 2 \Longrightarrow 2$, $2 + 0 \Longrightarrow 0$.

Weiterhin typisch für kleine Populationen ist, daß die zum Schluß dominierenden Spezies praktisch schon von Anfang an vorhanden sind (Abb. 4.1). Wenn sie nicht schon in der Anfangspopulation auftreten, so entstehen sie in den ersten Generationen. Der Reaktor ist nicht sonderlich innovativ.

Die Ähnlichkeit der vier Spezies ist ein Phänomen, daß bei allen Experimenten der Versuchsreihen A bis C aufgetreten ist. Bei keinem Versuch konnte nach 1000 Generationen zwei grundlegend unterschiedliche Spezies-Typen in einer signifikant hohen Konzentration beobachtet werden.

4.1.2 Verhalten einer mittleren Population ($M = 1000$)

In der folgenden Tabelle ist die Diversität der letzten Generation für alle 10 Runs der Populationsgröße $M = 1000$ dargestellt:

Run	1	2	3	4	5	6	7	8	9	10
Diversität der Gen. 999	1	2	1	1	24	1	1	1	8	8

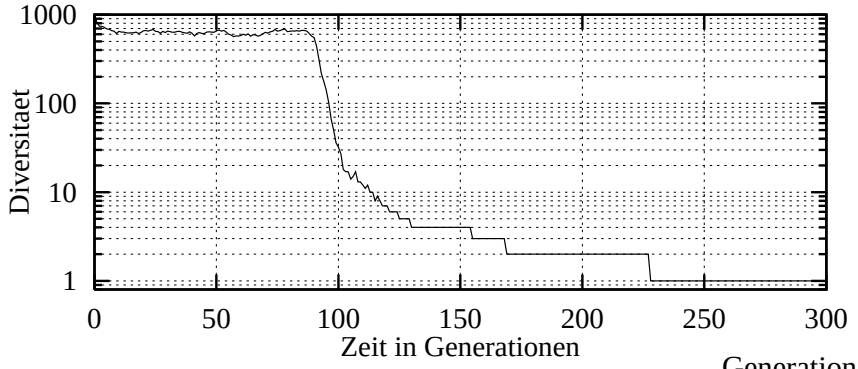
Die Anzahl der Läufe, in denen nur eine Spezies überlebt, ist häufiger als bei den Versuchen mit der kleinen Populationsgröße $M = 100$. Besteht eine am Ende dominierende Organisation aus mehreren Spezies, so ist diese tendenziel größer als für den Fall $M = 100$.

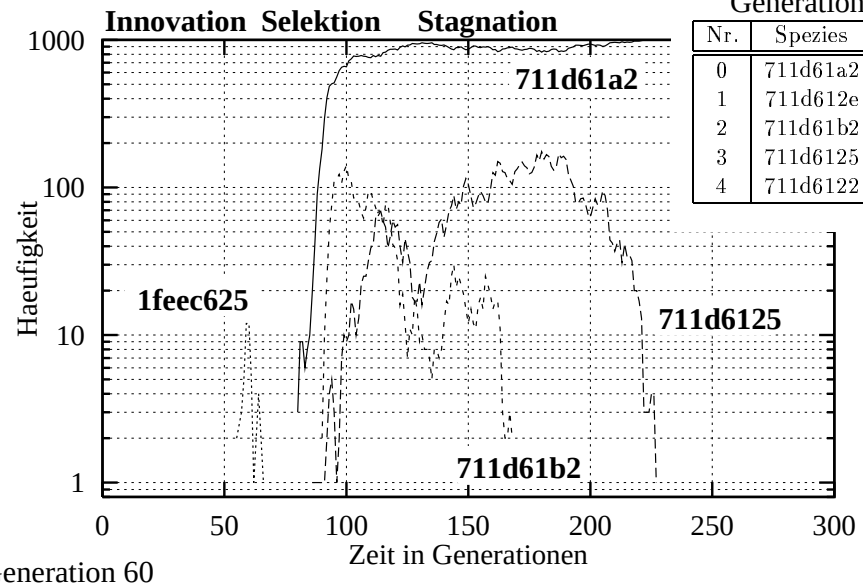
Die Abbildung 4.2 zeigt einen typischen Lauf, bei dem nur eine Spezies überlebt (Run 7). Betrachtet man den Verlauf der Diversität, so erkennt man drei Phasen:

In der **innovativen Phase** herrscht eine hohe Diversität. Laufend werden neue Spezies gebildet, die nicht in der Population vertreten sind. Es sind kaum Spezies vorhanden, die sich in einer signifikanten Konzentration von den anderen Spezies im Reaktor unterscheiden. Die in der Abbildung dargestellte Spezies `1feec625` ist die Spezies mit der höchsten Konzentration in dieser Phase. Die Reaktionsmatrix der Generation 60 zeigt, daß nur eine sehr lose Kopplung zwischen den Spezies besteht.

Die **selektive Phase** beginnt mit dem Auftauchen des fleißigen Replikators `711d61a2`. Dieser repliziert sich mit jeder anderen Spezies, falls er bei einer Kollision als Operator fungiert. Es findet eine Verdrängung der anderen Spezies statt. Gleichzeitig entstehen eine Reihe von Verwandten des fleißigen Replikators, die ähnliche oder sogar dieselben Eigenschaften zur Replikation aufweisen (s. Reaktionsmatrix der Generation 128, Abb. 4.2). Am Ende der selektiven Phase ist die Organisation der Spezies abgeschlossen. Es können keine neuen Spezies gebildet werden.

Es folgt die **stagnierende Phase** in der nichts Neues entsteht. Nur wenige, sehr ähnliche Spezies sind vorhanden.





Generation 128

Nr.	Spezies	0	1	2	3	4
0	711d61a2	0	0	0	0	0
1	711d612e	1	1	1	1	1
2	711d61b2	2	2	2	2	2
3	711d6125	3	3	3	3	3
4	711d6122	4	1	4	3	4

Generation 60

Nr.	Spezies	0	1	2	3	4	5	6	7	8	9	...40
0	1feec625	..	..	..	..	..	..	..	40	..	..	...
1	60ecc126	0	1	..	..	4	..	..	..	8	..	
2	645d8420	..	1	2	3	4	5	6	7	8	5	
3	ec81228a	0	..	..	25	..	9	..	..	..	9	
4	019d6824	..	..	..	..	4	..	..	..	8	..	
5	00e0d1ee	..	..	..	..	..	5	..	7	..	5	
6	81ac2380	..	1	2	3	4	5	6	7	8	5	
7	9fdd60ee	..	1	..	..	..	5	..	7	8	5	
8	819d6826	..	1	..	..	..	5	..	7	8	9	
9	00e0d1ef	..	..	..	..	..	5	..	7	..	9	...
...	...											...

Generation 88

Nr.	Spezies	0	1	2	3	4	5	6	7	...40
0	711d61a2	0	0	0	0	0	0	0	0	...
1	711d61a3	1	1	2	36	4	5	..	7	
2	01ac2225	..	..	15	..	..	..	..	..	
3	645d4802	0	1	2	3	4	5	6	7	
4	1e600819	1	1	2	36	4	5	..	7	
5	91bd6829	..	..	..	..	..	..	..	..	
6	10dd6820	..	..	..	..	..	..	6	..	
7	71ac230d	1	1	..	36	..	..	..	..	...
...	...									...

Abb. 4.2: *Typischer Run der Versuchsreihe A mit einer mittleren Populationsgröße. Das Symbol ".." in einer Reaktinsmatrix bedeutet, daß das Reaktionsprodukt an dieser Stelle nicht in der Matrix vorkommt. Hier erfassen die Matrizen der Generationen 60 und 88 die 41 häufigsten Spezies.*
$M = 1000$, Kode I, Replikation erlaubt, Run 7. Erläuterung im Abschnitt 4.1.2.

Der Zustand des Reaktors ab Generation 132 (Abb. 4.2) ist nicht (asymptotisch) stabil. Die Population befindet sich in einem indifferenten Gleichgewicht. Ihre Reaktionsmatrix hat die Form:

Nr.	0	1	...	n
0	0	0	...	0
1	1	1	...	1
...	...	...	...	...
n	n	n	...	n

Gilt eine derartige **indifferente Matrix**, so ist für jede *anwesende* Spezies die Wahrscheinlichkeit, daß ihre Häufigkeit um eins zunimmt, gleich der Wahrscheinlichkeit, daß sie um eins abnimmt, und zwar unabhängig vom Zustand des Reaktors. Das entsprechende kontinuierliche DGL-Modell nach Abschnitt 2.2.3 hat ausschließlich konstante Lösungen, so daß jeder Zustand stationär ist.

Der diskrete algorithmische Reaktor hat einen wichtigen qualitativen Unterschied zu seinem kontinuierlichen DGL-Modell. Spezies können aufgrund der stochastischen Schwankungen ihrer Häufigkeiten aussterben. In dem in der Abbildung 4.2 dargestellten Lauf bleibt am Ende nur eine Spezies übrig.

Anmerkung 4.1.1 *Auch bei einer stabilen Organisation wie in Abbildung 4.1 können theoretisch Spezies aussterben. Die Wahrscheinlichkeit ist aber gegenüber indifferenten Organisationen vernachlässigbar gering.*

4.1.3 Verhalten einer großen Population ($M = 10^4$)

Für die Populationsgröße $M = 10^4$ sind nur 5 Läufe durchgeführt worden, da keine qualitative Variation aufgetreten ist.

In der folgenden Tabelle ist die Diversität der letzten Generation für alle 5 Runs dargestellt:

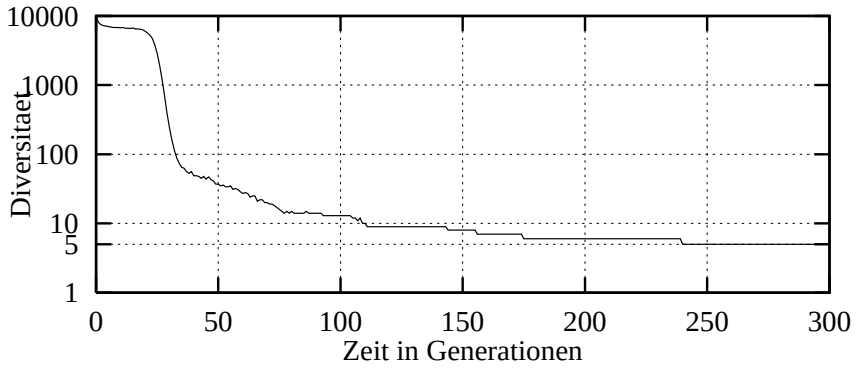
Run	1	2	3	4	5
Diversität der Gen. 999	$2 \rightarrow 1$	$5 \rightarrow 1$	$6 \rightarrow 1$	1	$5 \rightarrow 1$

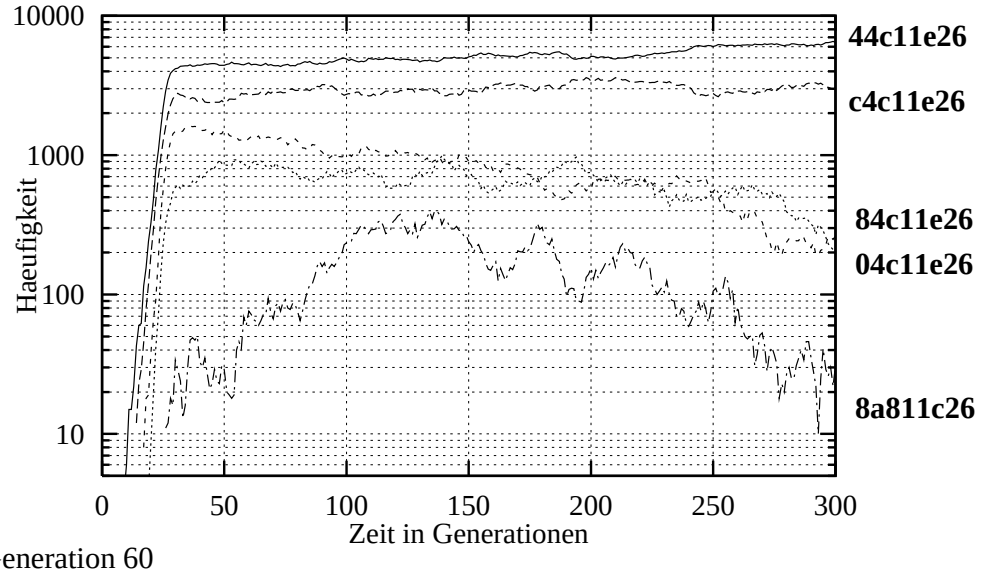
Die Schreibweise $5 \rightarrow 1$ deutet an, daß zwar 5 verschiedene Spezies vorhanden sind, diese sich aber in einem indifferenten Gleichgewichtszustand befinden, so daß bei einer längeren Simulationszeit nur eine Spezies überleben wird.

Die Abbildung 4.3 zeigt den typischen Verlauf einer Simulation (Run 1). Die innovative Phase hoher Diversität ist deutlich kürzer. Dies ist darauf zurückzuführen, daß mit einer hohen Wahrscheinlichkeit fleißige Replikatoren schon in der Anfangspopulation vorhanden sind (ca. 50%, vgl. Abschnitt 3.6.5).

Schnell bildet sich eine indifferente Organisation fleißiger Replikatoren (s. Reaktionsmatrix der Generation 60, Abb. 4.3).

Bei Experimenten mit Populationen größer als 10^4 hat sich keine neue qualitative Dynamik ergeben. Die indifferente Organisation bildet sich nur früher aus, und die innovative Phase verkürzt sich.





Generation 60

Nr.	Spezies	0	1	2	3	4	5	6	7	8	9	10	...
0	44c11e26	0	0	0	0	0	0	0	0	0	0	0	...
1	c4c11e26	1	1	1	1	1	1	1	1	1	1	1	
2	04c11e26	2	2	2	2	2	2	2	2	2	2	2	
3	84c11e26	3	3	3	3	3	3	3	3	3	3	3	
4	44c11a26	4	4	4	4	4	4	4	4	4	4	4	
5	24c11e26	5	5	5	5	5	5	5	5	5	5	5	
6	8a811c26	6	6	6	6	6	6	6	6	6	6	6	
7	45011c26	7	7	7	7	7	7	7	7	7	7	7	
8	44c11c26	8	9	14	15	8	23	6	7	8	9	10	
9	c4c11c26	8	9	14	15	8	23	6	7	8	9	10	
10	84511c26	10	10	10	10	10	10	10	10	10	10	10	...
...	...												...

Generation 88

Nr.	Spezies	0	1	2	3	4
0	44c11e26	0	0	0	0	0
1	c4c11e26	1	1	1	1	1
2	84c11e26	2	2	2	2	2
3	04c11e26	3	3	3	3	3
4	8a811c26	4	4	4	4	4

Abb. 4.3: *Typischer Run der Versuchsreihe A mit einer großen Population.*
$M = 10^4$*, Kode I, Replikation erlaubt, Run 1. Erläuterung im Abschnitt 4.1.3.*

4.2 Versuchsreihe B – Simulationen *ohne* Replikation

Bei den Experimenten der Versuchsreihe A dominieren die fleißigen Replikatoren. Ihr Auftauchen bewirkt eine schnelle Verarmung der Vielfalt.

Um dem entgegenzuwirken, könnte man die fleißigen Replikatoren verbieten, entsprechend der Behandlung des Destruktors im Algorithmus 2.3.1. Dagegen spricht: 1.) Es sind nicht alle fleißigen Replikatoren bekannt, so daß ein effizientes Kriterium zu deren Erkennung gefunden werden muß. 2.) Ein fleißiger Replikator ist im Gegensatz zu dem String 0000 eine interessante und komplexe Spezies.

In der Versuchsreihe B wird dagegen, wie in [FONTANA 1991], die Replikation unterdrückt, indem alle Kollisionen, die zu einer Replikation führen, als elastisch gelten.

Für die neue Meßgröße "Produktivität" ist es nützlich, deren durchschnittlichen Wert für eine zufällig generierte Population zu kennen: Die Wahrscheinlichkeit, daß die Reaktion zwischen zwei zufällig erzeugten Spezies eine Replikation ist, beträgt $31.5 \pm 0.1\%$. Damit liegt die Produktivität einer zufälligen Population näherungsweise bei 68 %.

4.2.1 Verhalten einer kleinen Population ($M = 100$)

In der folgenden Tabelle ist die Diversität und Produktivität Φ der letzten Generation dargestellt.

Run (Reihe B, $M = 100$)	1	2	3	4	5	6	7	8	9	10
Diversität in Gen. 999	8	4	4	16	29	2	4	3	2	2
Produktivität Φ in Gen. 999	55%	68%	0%	70%	74%	49%	75%	0%	52%	51%

Die Abbildung 4.4 zeigt den Run 6 als einen typischen Lauf. Die durchschnittliche Vielfalt ist höher als in der Versuchsreihe A. Allerdings reduziert sich die Diversität genau wie dort. Die überlebenden Spezies sind praktisch von Anfang an vorhanden. Die Produktivität sinkt leicht unter die einer zufälligen Population.

4.2.2 Verhalten einer mittleren Population ($M = 1000$)

Die 10 Läufe mit einer Populationsgröße von $M = 1000$ im Überblick:

Run (Reihe B, $M = 1000$)	1	2	3	4	5	6	7	8	9	10
Diversität in Gen. 999	7	47	8	58	11	18	26	98	419	549
Produktivität Φ in Gen. 999	67%	57%	68%	76%	0.3%	0.3%	88%	75%	74%	58%

Wie für den Fall $M = 100$ findet keine Innovation statt. Es wird nur das verstärkt, was von Anfang an vorhanden ist. Die sich ausbildenden Organisationen sind zwar größer, qualitativ aber wie im Falle $M = 100$ strukturiert.

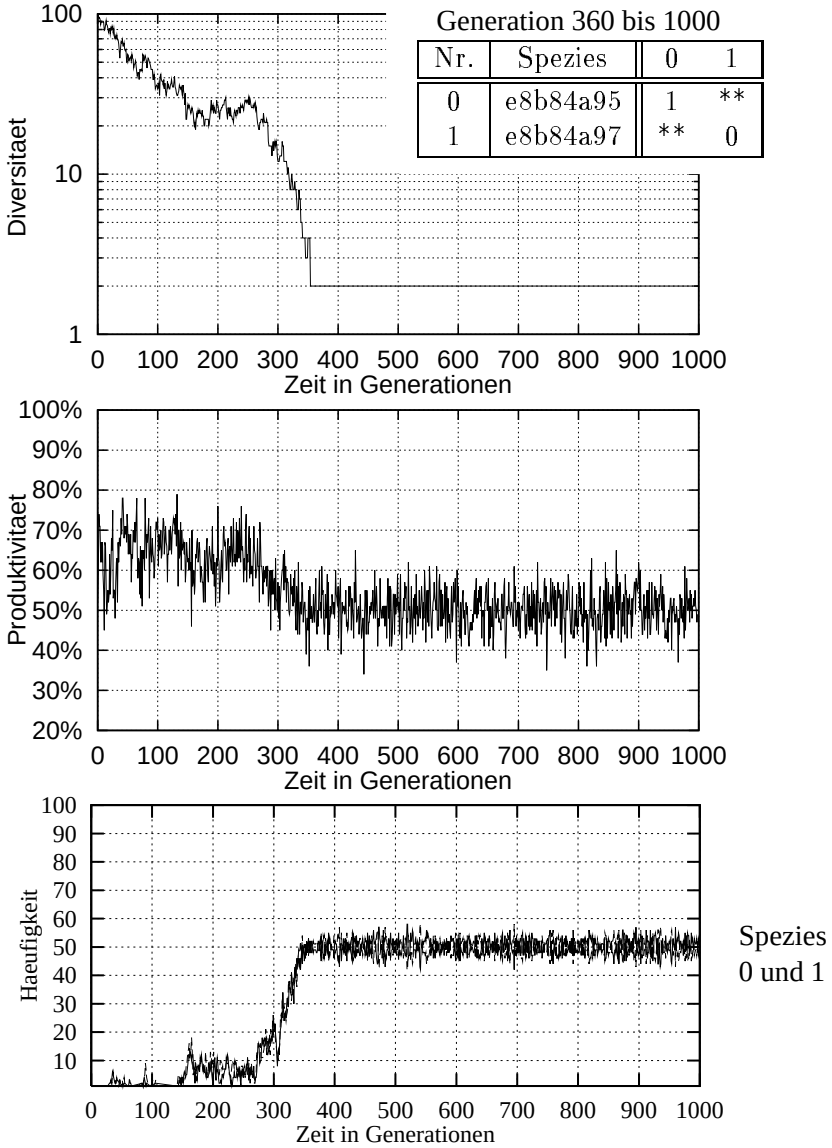


Abb. 4.4: *Typischer Run der Versuchsreihe B mit einer kleinen Populationsgröße. Das Symbol "**" in der Reaktionsmatrix deutet eine elastische Wechselwirkung aufgrund des Replikationsverbots an.*
$M = 100$, *Replikation verboten, Kode I, Run 6. Erläuterung in Abschnitt 4.2.1.*

4.2.3 Verhalten einer großen Population ($M = 10^4$)

Mit wachsender Populationsgröße läßt sich ein Phasenübergang im qualitativen Verhalten der Dynamik ausmachen. Die folgende Übersicht zeigt, daß die Größe einer verbleibenden Organisation sich nicht nennenswert von denen kleinerer Populationen unterscheidet. Jedoch liegt deren Produktivität deutlich über der einer zufälligen Population.

Run	1	2	**3**	4	5	6	**7**	8	9	10
Diversität in Gen. 999	12	12	**2412**	481	32	26	**42**	34	8	18
Produktivität Φ in Gen. 999	92%	91%	**80%**	82%	95%	96%	**97%**	96%	88%	94%

Der 3. Run stellt eine seltene Ausnahme dar und wird gesondert besprochen.

Ein typischer Verlauf ist in der Abbildung 4.5 dargestellt. Er beginnt mit einer **produktiven Phase**, die folgendermaßen charakterisiert ist: 1.) Die Diversität ist hoch. 2.) Die Produktivität entspricht der einer zufälligen Population. 3.) Es entstehen laufend neue Spezies. 4.) Keine Spezies ist außergewöhnlich häufig.

Am Ende der innovativen Phase taucht eine Gruppe von Spezies auf, die für sich genommen eine höhere Produktivität besitzen als die übrigen. Sie setzen sich in der selektiven Phase durch und bilden eine abgeschlossene Organisation (s. Reaktionsmatrix der Abb. 4.5).

Die Organisation besteht aus kleinen Gruppen von kooperierenden Spezies, die sich innerhalb der Gruppe gegenseitig erzeugen. Zum Beispiel reagiert der String `aa111f26` (Nr. 0) unabhängig von dem gewählten Operanden zu dem String spea111c26 (Nr. 1) und umgekehrt.

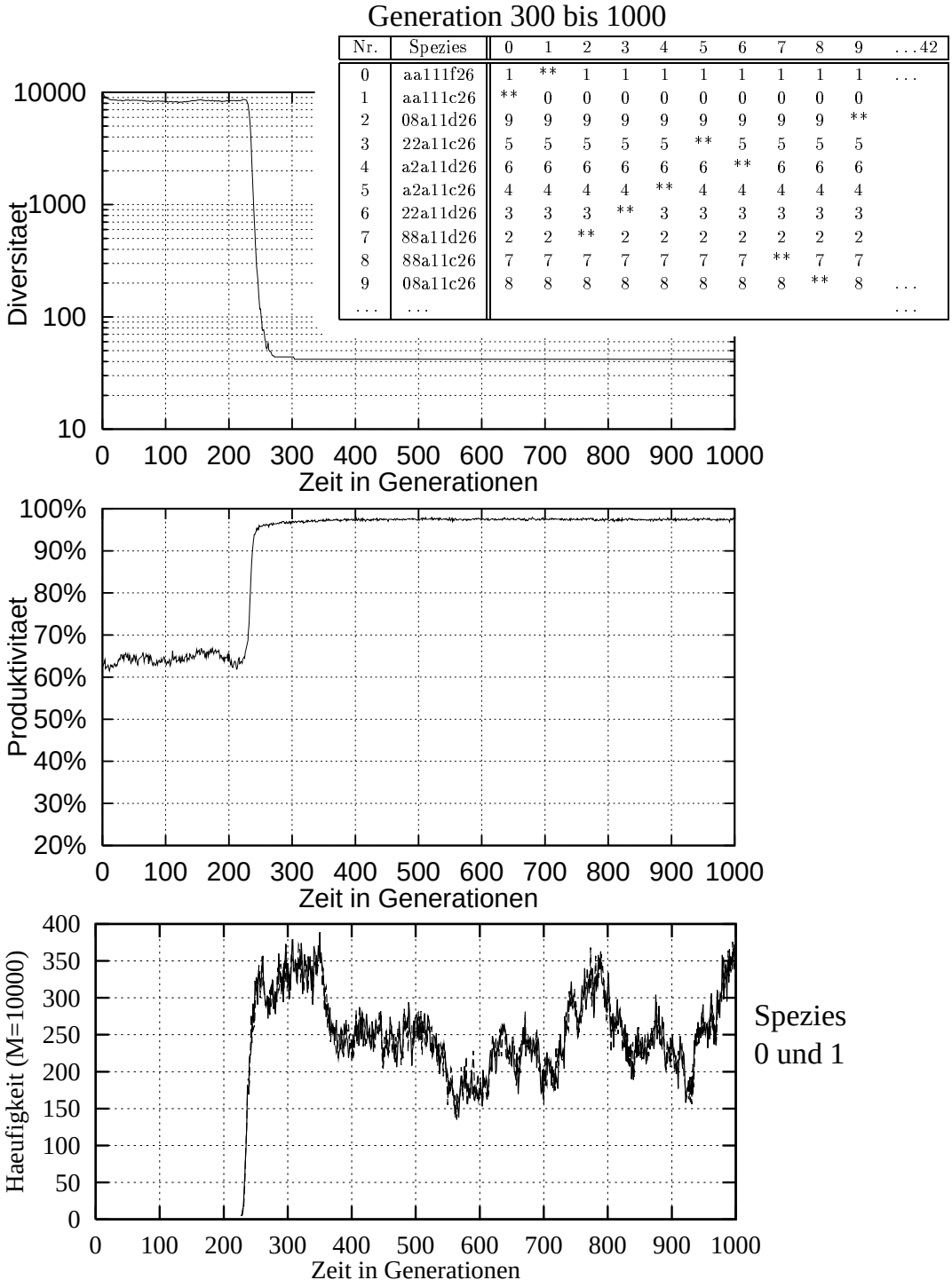
Wird ein solches Paar bzw. eine Gruppe als eine **Quasispezies** aufgefaßt, so entsprechen die Reaktionsbeziehungen zwischen diesen Quasispezies denen der indifferenten Organisationen, die sich bei großen Populationen der Versuchsreihe A gebildet haben. Die Reaktionsmatrix auf der Ebene der Quasispezies hat die gleiche Form wie die Reaktionsmatrix dort. Dem entsprechend kann man die Reaktionsmatrix der Abbildung 4.5 kompakter darstellen:

Quasispezies	0/1	2/7/8/9	3/4/5/6	...
0/1	0/1	0/1	0/1	...
2/7/8/9	2/7/8/9	2/7/8/9	2/7/8/9	...
3/4/5/6	3/4/5/6	3/4/5/6	3/4/5/6	...
...	...	...	...	...

Außergewöhnliches Verhalten im 3. Run

Der 3. Run zeigt ein außergewöhnliches Verhalten, das eigentlich typisch für die nächste Versuchreihe C ist.

Das Diagramm der Diversität ist deutlich komplexer (Abb. 4.6). An die Stelle der stagnierenden Phase ist eine Phase relativ hoher Diversität getreten, die stark zwischen 1800 und 2500 Spezies schwankt.



Nr.	Spezies	0	1	2	3	4	5	6	7	8	9	...42
0	aa111f26	1	**	1	1	1	1	1	1	1	1	...
1	aa111c26	**	0	0	0	0	0	0	0	0	0	
2	08a11d26	9	9	9	9	9	9	9	9	9	**	
3	22a11c26	5	5	5	5	5	**	5	5	5	5	
4	a2a11d26	6	6	6	6	6	6	**	6	6	6	
5	a2a11c26	4	4	4	4	**	4	4	4	4	4	
6	22a11d26	3	3	3	**	3	3	3	3	3	3	
7	88a11d26	2	2	**	2	2	2	2	2	2	2	
8	88a11c26	7	7	7	7	7	7	7	**	7	7	
9	08a11c26	8	8	8	8	8	8	8	8	**	8	...
...	...											...

Abb. 4.5: *Typischer Run der Versuchsreihe B mit einer großen Population.*
$M = 10^4$*, Replikation verboten, Kode I, Run 7. Erläuterung in Abschnitt 4.2.3.*

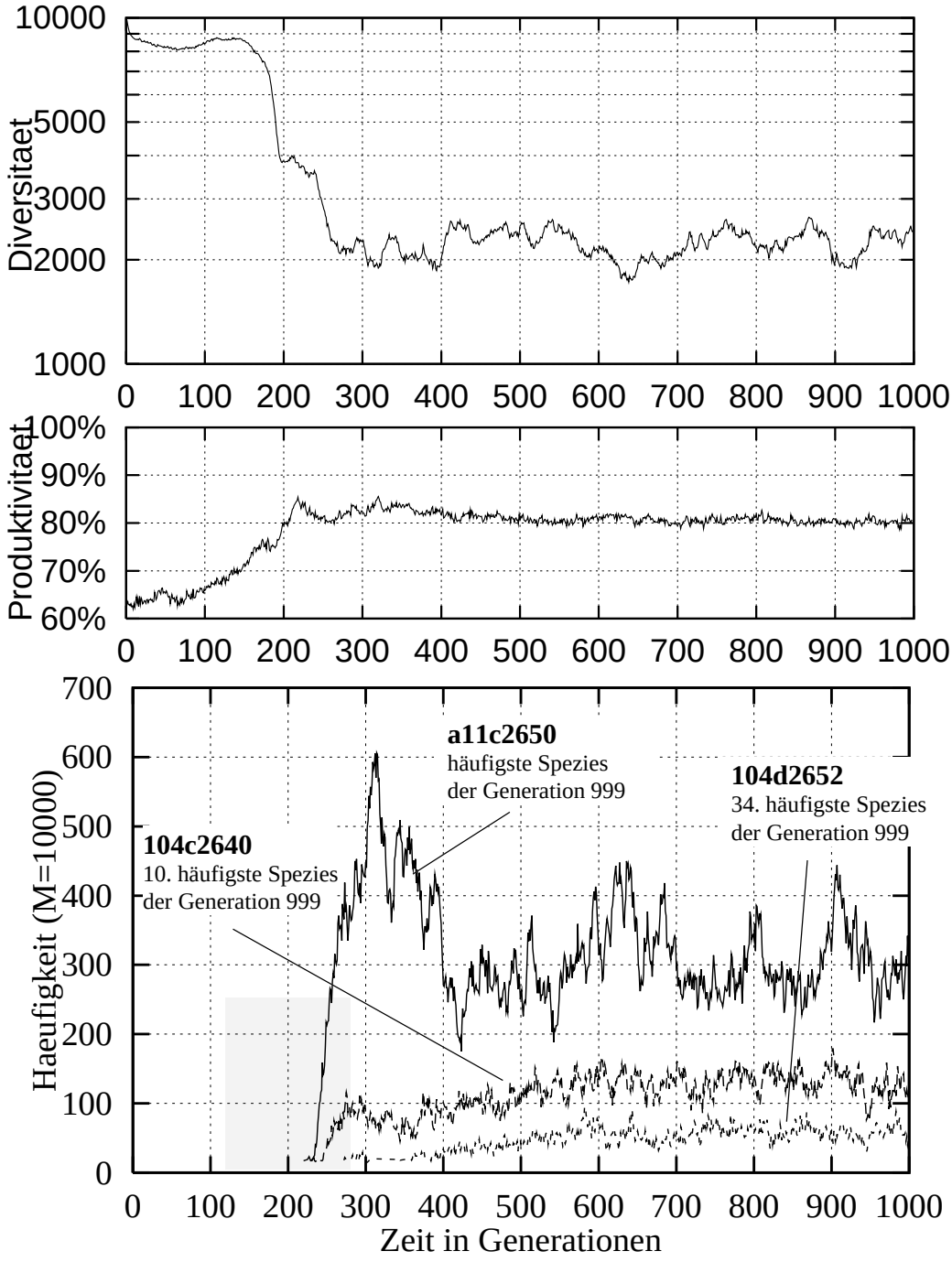


Abb. 4.6: *Außergewöhnlicher Run der Versuchsreihe B. Die Dynamik ist deutlich komplexer. Es gibt eine evolutive Phase (grauer Bereich), die vergrößert in der folgenden Abbildung 4.7 dargestellt ist.*
$M = 10^4$, Replikation verboten, Kode I, Run 3.

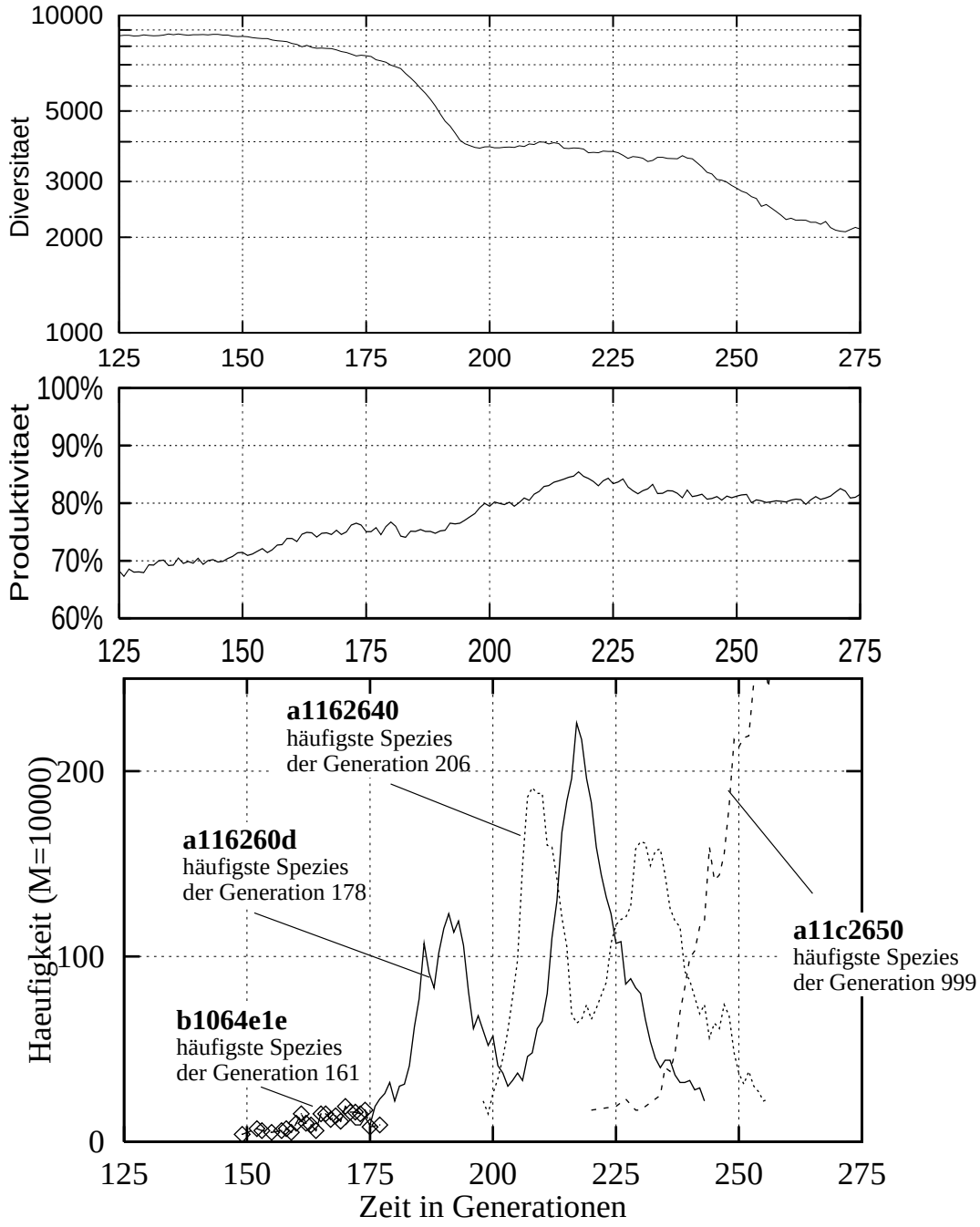


Abb. 4.7: *Evolutive Phase des atypischen 3. Runs der Versuchsreihe B. Diese ist dadurch gekennzeichnet, daß Spezies mit relativ hohen Konzentrationen auftauchen und wieder verschwinden. Die am Ende dominierende Spezies entsteht nicht "auf einen Schlag" sondern ist das Produkt eines Entwicklungsprozesses, der in mehreren Schritten abläuft.*
$M = 10^4$, Replikation verboten, Kode I, Run 3.

Nr.	Spezies	Häufigkeit	0	1	2	3	4	5	6	7	8	9	...	90	91	92	93	94	95	96	97	98	99
0	a11c2650	313	2	**	**	2	**	**	1	2	1	5		..	4	2	2	..	2	1	2	..	1
1	104d2650	241	12	**	**	8	**	**	**	21	**	6		12	76	..	57	8	..	34	23	35	**
2	a11d2650	227	**	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
3	a11c2640	189	10	10	10	20	10	10	20	10	20	20		10	67	95	31	20	**	67	31	31	..
4	14072650	160	25	**	**	17	**	**	11	62	17	11		25	17	..	62	17	..	11	**	25	..
5	a11f2650	144	2	**	**	2	29	2	1	2	1	2		..	29	2	2	..	2	1	2	..	1
6	104d2640	137	3	**	13	**	11	9	**	13	**	**		94	17	3	13	**	3	**	13	94	**
7	a11d26d0	124	**	**	**	0	**	**	1	2	12	5		**	25	0	2	90	0	1	2	90	1
8	104c2640	116	3	6	13	**	11	9	**	..	**	**		94	17	3	..	**	20	6	13	94	6
9	a11f2640	115	19	19	19	26	19	19	26	64	26	26		19	..	..	..	26	..	..	54	54	..
90	a11a2650	18	2	**	**	2	**	**	1	2	1	5		..	4	2	2	..	2	1	2	..	1
91	14062646	17	73	**	45	88	**	..	**	..	**	..		..	**	..	..	..	..	**	..	..	**
92	a11c2600	17	..	83	..	..	..	..	83	..	83	..		..	..	..	..	..	..	83	..	..	83
93	a11d26d6	17	..	52	68	..	33	..	52	68	..	..		..	65	..	68	..	..	52	68	..	52
94	a11a2640	17	..	..	..	..	..	..	..	..	..	..		..	..	..	..	..	..	..	..	..	..
95	a11c2680	17	**	**	**	**	**	**	**	**	**	**		**	**	**	**	**	**	**	69	**	**
96	104d2646	16	..	**	..	..	..	..	**	..	..	..		..	**	..	..	..	..	**	..	..	**
97	a11d2655	16	..	..	..	..	..	..	..	..	..	..		..	..	..	..	..	..	..	..	..	..
98	a11a2654	16	..	..	..	..	..	..	..	..	..	..		..	..	..	..	..	..	..	..	..	..
99	104d2648	16	**	**	**	**	**	**	**	**	**	**		**	**	**	**	**	**	**	**	**	**

Abb. 4.8: *Reaktionsmatrix zur Abbildung 4.6. Es handelt sich um einen Ausschnitt der Reaktionsmatrix der 100 häufigsten Spezies in der Generation 999. Versuchsreihe B, Run 3.*

Typisch für die vorhergehenden Experimente ist, daß die ersten Spezies mit hohen Konzentrationen auch in der stagnierenden Phase häufig sind, und nur eventuell langsam aussterben, weil andere gleichwertige Spezies vorhanden sind. Sie entstehen praktisch "mit einem Schlag", reduzieren wegen ihrer Selbstvermehrung die Vielfalt und beenden damit die innovative Phase.

Hier dagegen läuft der Entstehungsprozeß der endgültigen Organisation kontinuierlicher ab. Es gibt eine so genannte **evolutive Phase**, in der Spezies mit hoher Konzentration auftauchen und wieder aussterben. Die Abbildung 4.7 zeigt die evolutive Phase des 3. Runs.

Methoden zur Umgehung des Replikationsverbots

Das "Standardverfahren", das sich in den Simulationen mit $M >= 10^4$ entwickelt hat und das auch die Spezies des 7. Runs anwenden, enthält zwei Schritte:

1. Das Kopieren des Operatorregisters in das Operandenregister.
2. Das Einbringen einer geringen Variation durch die Invertierung von wenigen Bits im Operandenregister mittels der NOT-Operation. Die resultierende Spezies bewirkt dasselbe und macht diese Variation wieder rückgängig, falls sie als Operator bei einer Reaktion fungiert.

Spezies	`aa111f26`
Kode	Operation
0110	CPON
0010 1111	SETP 1111
0001	MOV
0001	MOV
0001	MOV
1010	NOT
1010	NOT

Spezies	`aa111c26`
Kode	Operation
0110	CPON
0010 1100	SETP 1100
0001	MOV
0001	MOV
0001	MOV
1010	NOT
1010	NOT

Zustand des Operandenregisters während der Reaktion eaa111f26 + $\mathbf{s}' \Longrightarrow$ aa111c26 :

Operandenregister nach dem 3. MOV:	1010 1010 0001 0001 0001 111*1* 0010 0110
Operandenregister nach dem 2. NOT:	1010 1010 0001 0001 0001 11**00** 0010 0110

Abb. 4.9: *Beispiel für die Standardmethode zur Umgehung des Replikationsverbots. Das eigene Bitmuster wird zunächst in das Operandenregister kopiert und dann mit der NOT-Operation an wenigen Stellen (hier an zwei Stellen) invertiert. Die Variation wird so angebracht, daß die resultierende Spezies die gleiche Funktionalität aufweist und so bei ihrer späteren Anwendung als Operator diese Variation rückgängig macht. Die dargestellten Programme stammen von den beiden häufigsten Spezies der 999. Generation im 7. Run (vgl. Reaktionsmatrix der Abb. 4.5).*

Die Abbildung 4.2.3 zeigt ein Beispiel aus dem 7. Run. Die Dominanz der NOT-Funktion unter den arithmetischen Operationen wird in dem Befehlshistogramm (Abb. 4.12) deutlich. Im Vergleich mit der Versuchsreihe A (Abb. 4.10) läßt sich ein deutlicher Unterschied in der Zusammensetzung der Spezies erkennen. Ist bei einem bestehenden Replikationsverbot die NOT-Operation ein wichtiger Bestandteil, so ist sie in der Versuchsreihe A nur hinderlich und wird dort – bei nicht zu kleinen Populationen – vollständig verdrängt.

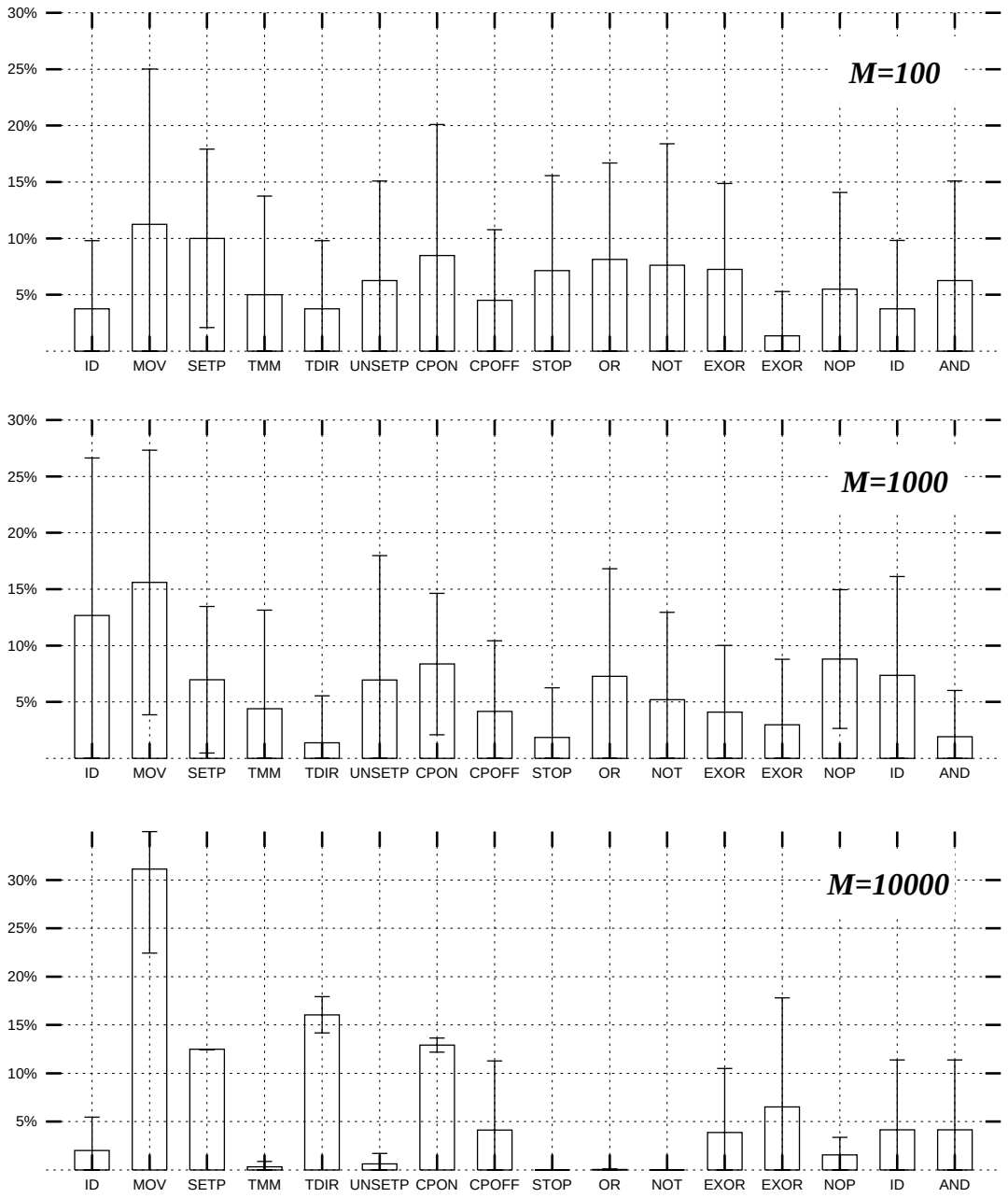


Abb. 4.10: *Histogramme der Befehle zu der Versuchsreihe A. Dargestellt ist die relative Verteilung der Befehle in der Generation 990, gemittelt über 10 Runs. Ferner ist die Standardabweichung der Einzelmessung eingetragen. Bei einer kleinen Population ist die Streuung sehr groß, da die schnelle Reduzierung der Diversität vielfältigen Spezies eine Chance zum Überleben bietet. Läufe mit* $M = 10^4$ *werden dagegen von fleißigen Replikatoren dominiert, die Spezies, wie sie bei kleineren Populationen übrig bleiben, verdrängen. Das Befehlshistogramm für* $M = 10^4$ *spiegelt die durchschnittliche Zusammensetzung eines fleißigen Replikators wieder.*

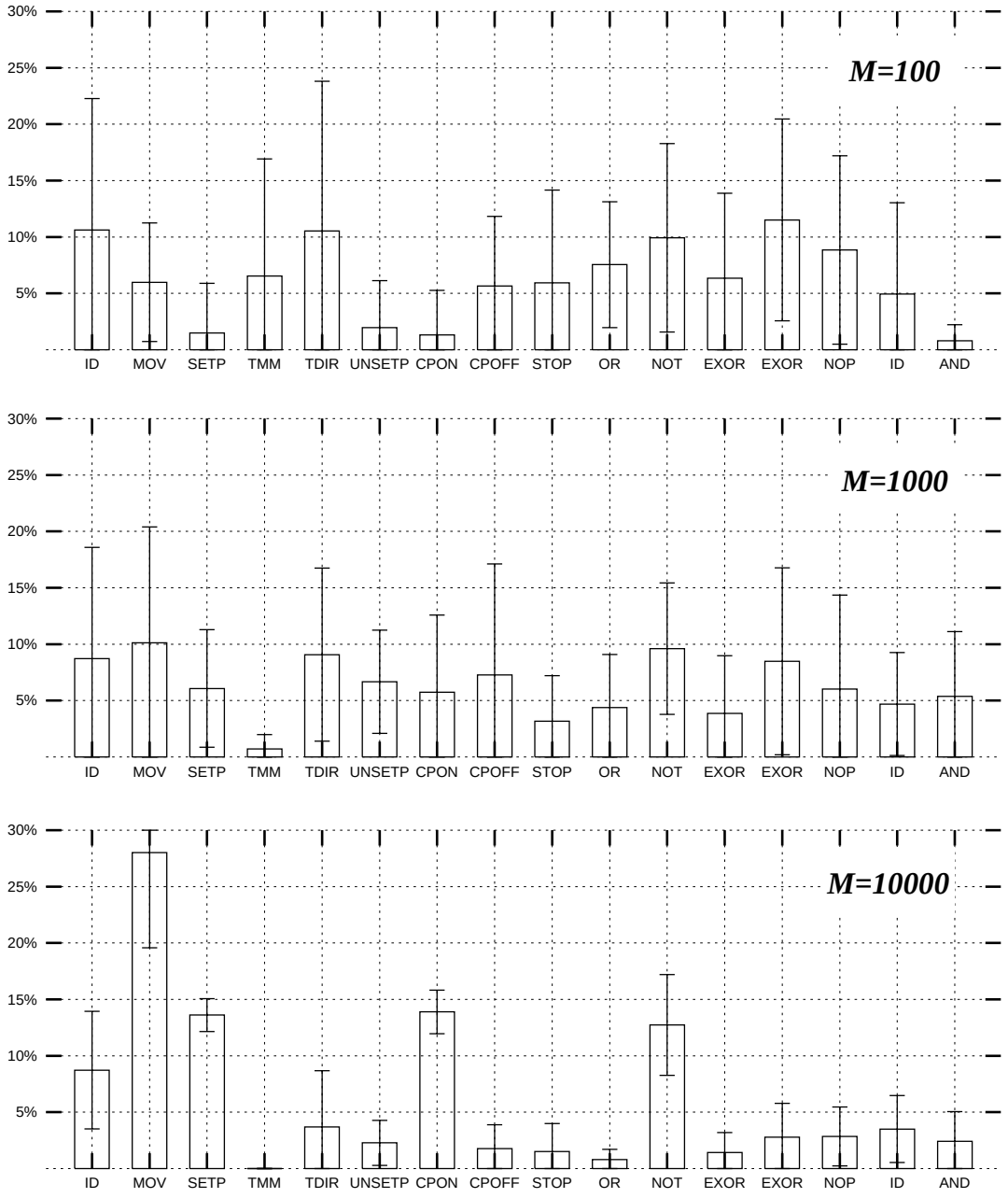


Abb. 4.11: *Histogramm der Befehle zu der Versuchsreihe B für die Generation 990, gemittelt über jeweils 10 Runs. Für kleine Populationen gilt das gleiche wie für die Versuchsreihe A (Abb. 4.10). Die typische Spezies, die sich in großen Populationen entwickelt, enthält eine oder mehrere NOT-Operationen, um das Replikationsverbot zu umgehen (vgl. Abb. 4.2.3).*

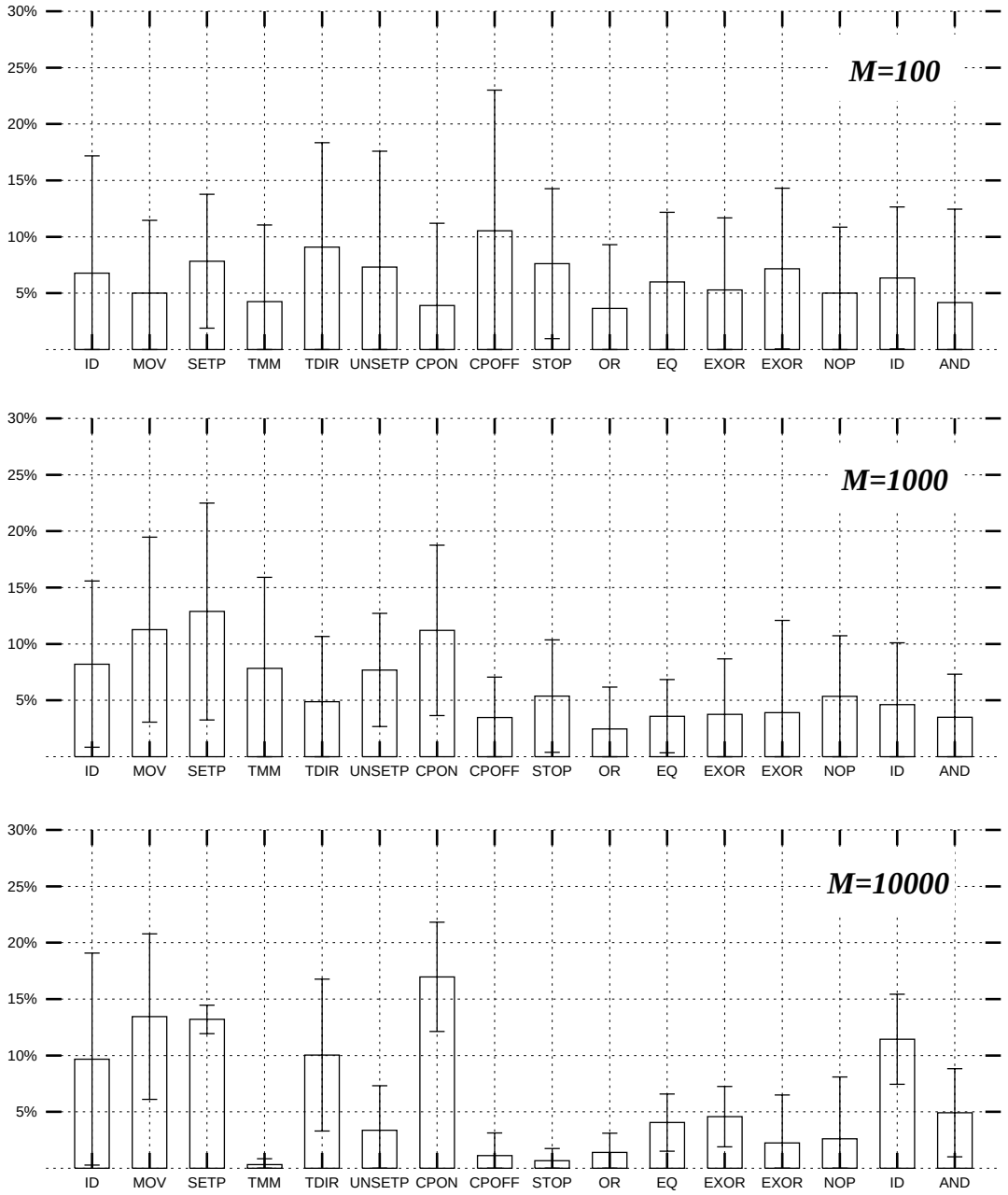


Abb. 4.12: *Histogramm der Befehle zu der Versuchsreihe C für die Generation 990, gemittelt über jeweils 10 Läufe. Der Befehlssatz enthält anstatt der NOT-Operation die EQ-Operation (Äquivalenz).*

4.3 Versuchsreihe C – *ohne* Replikation und *ohne* NOT-Operation

Das in der Versuchsreihe B eingeführte Replikationsverbot konnte relativ leicht mittels der NOT-Funktion umgangen werden. In der Versuchsreihe C wird die NOT-Funktion entfernt und durch die Äquivalenz (EQ) bzw. Identität (ID) ersetzt. Dadurch wird der Schwierigkeitsgrad erhöht, eine eng zusammenhängende, selbsterhaltende Organisationsstruktur zu bilden. Ferner sollen in dieser Reihe sehr große Populationen ($M = 10^6$) betrachtet werden.

Der Vollständigkeit halber folgt die Zusammenfassungen der Experimente mit $M = 100, M = 1000$ und $M = 10^4$ (Kode II):

Run (Reihe C, $M = 100$)	1	2	3	4	5	6	7	8	9	10
Diversität in Gen. 999	2	5	4	8	2	7	3	3	4	8
Produktivität Φ in Gen. 999	58%	0%	39%	49%	0%	0%	0%	0%	51%	68%
Run (Reihe C, $M = 1000$)	1	2	3	4	5	6	7	8	9	10
Diversität in Gen. 999	16	645	81	158	40	834	59	45	25	20
Produktivität Φ in Gen. 999	79%	63%	78%	82%	62%	60%	74%	70%	0.2%	5%
Run (Reihe C, $M = 10^4$)	1	2	3	4	5	6	7	8	9	10
Diversität in Gen. 999	??	282	32	14	1176	62	274	92	108	719
Produktivität Φ in Gen. 999	??	86%	61%	44%	86%	98%	75%	61%	79%	81%

Durch die Modifikation des Befehlssatzes sind die Werte der Produktivität nur schwer mit denen der Reihen A und B vergleichbar. Wird nämlich NOT durch ID ersetzt (Kode III), so liegt die durchschnittliche Produktivität nach 1000 Generationen deutlich höher ($93\% \pm 7\%$) als bei einer Ersetzung von NOT durch EQ ($80\% \pm 15\%$) – für $M \geq 10^4$.

Typisch für die Reihe C ist, daß sich die am Ende dominierende Organisation nicht plötzlich bildet, sondern das Ergebnis eines erkennbaren Entwicklungsprozesses ist. Dieses Phänomen wird in den folgenden beiden Abschnitten anhand zweier typischer Runs mit sehr großen Populationen gezeigt.

4.3.1 Komplexes Verhalten im Run C1 (Kode III, $M = 10^6$)

Bei diesem Experiment ist NOT durch ID ersetzt worden. Die simulierte Zeit beträgt 2100 Generationen. Zur Erstellung der Häufigkeitsdiagramme sind in jeder zweiten Generation die 200 häufigsten Spezies aufgezeichnet worden.

Die Abbildung 4.13 zeigt eine Übersicht über den gesamten Run. Auffällig ist, daß die Diversität in den Generationen 117 bis 126 um ca. 40000 Spezies zunimmt (Abb. 4.14). Dieses neue Phänomen kann wegen der sehr großen Population nicht auf die zufälligen Elemente des Reaktor-Algorithmus zurückgeführt werden, die sich in den Abbildungen 4.4 und 4.5 als Rauschanteil in der Produktivität und Diversität bemerkbar gemacht haben. Es muß statt dessen der Innovationsfähigkeit zugeschrieben werden.

Die am Ende dominierenden Spezies tauchen erst sehr spät auf (Abb. 4.13). Sie sind das Ergebnis eines in mehreren Stufen ablaufenden Entwicklungsprozesses, der exemplarisch anhand einiger Spezies in der Abbildung 4.14 aufgezeigt ist.

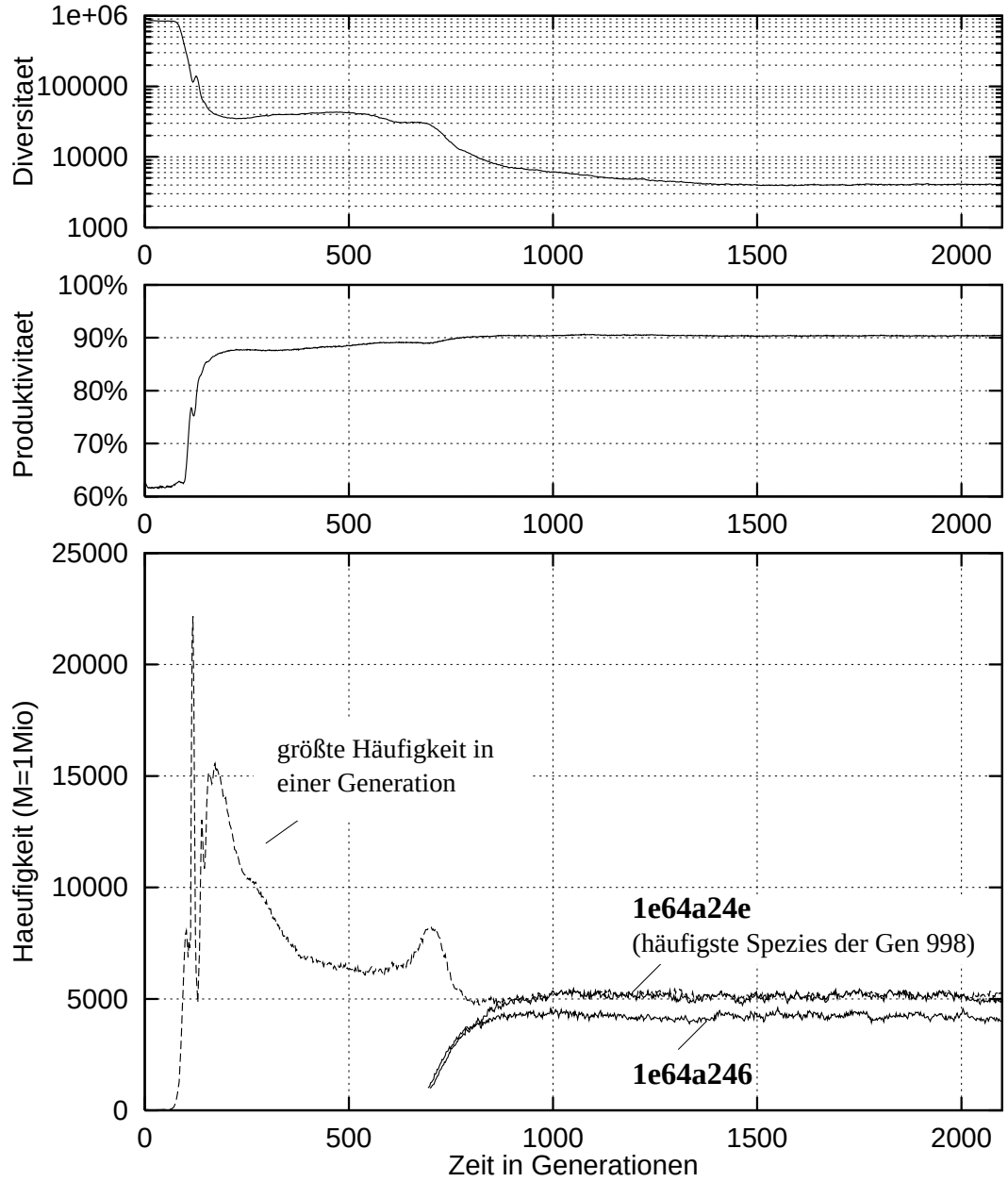


Abb. 4.13: *Typischer Run der Versuchsreihe C mit einer sehr großen Population von $M = 10^6$. Durch die Ersetzung der NOT-Operation durch ID kann das Replikationsverbot nicht mehr einfach mit der Standardmethode der Versuchsreihe B umgangen werden. Die am Ende dominierenden Spezies verwenden eine Crossover-Methode (Abb. 4.16). Sie sind das Produkt einer sehr langen evolutiven Phase (Abb. 4.14).*
$M = 10^6$, Replikation verboten, Kode III, Run C1.

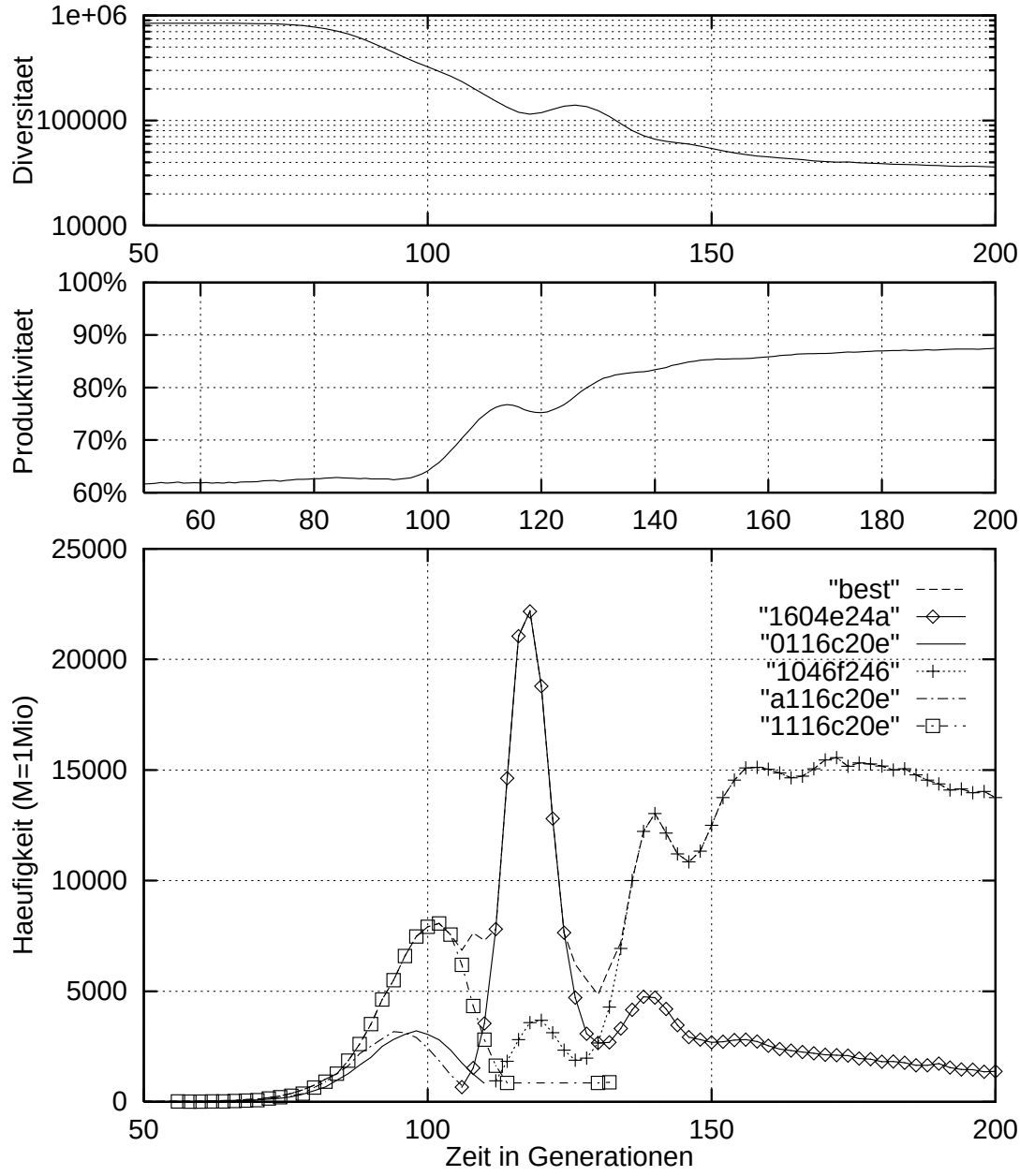


Abb. 4.14: *Ausschnittsvergrößerung der evolutiven Phase von Abbildung 4.13. Die dargestellten Spezies stehen stellvertretend für ganze Gruppen von Spezies, die mit hohen Konzentrationen auftauchen und wieder verschwinden.*
$M = 10^6$, Replikation verboten, Kode III, Run C1.

Nr.	Spezies	Häufigkeit	0	1	2	3	4	5	6	7	8	9	10	...	90	91	92	93	94	95	96	97	98	99
0	1e64a24e	5338	**	**	**	**	**	1	3	2	**	**	**		1	52	64	9	71	13	11	8	15	13
1	1064a24e	5277	**	**	**	**	0	**	3	2	**	**	**		**	52	64	9	71	13	11	8	15	13
2	1064b24e	4958	**	**	**	**	0	1	3	**	**	**	**		1	52	64	9	71	13	11	8	15	13
3	1e64b24e	4886	**	**	**	**	0	1	**	2	**	**	**		1	52	64	9	71	13	11	8	15	13
4	1e64a246	4456	**	5	7	6	**	**	**	**	24	19	20		5	**	**	19	**	23	18	24	29	23
5	1064a246	4423	4	**	7	6	**	**	**	**	24	19	20		**	**	**	19	**	23	18	24	29	23
6	1e64b246	4281	4	5	7	**	**	**	**	**	24	19	20		5	**	**	19	**	23	18	24	29	23
7	1064b246	4188	4	5	**	6	**	**	**	**	24	19	20		5	**	**	19	**	23	18	24	29	23
8	1a46e24e	3605	**	**	**	**	0	1	3	2	**	**	**		1	52	64	9	71	13	11	**	15	13
9	1064f24e	3596	**	**	**	**	0	1	3	2	**	**	**		1	52	64	**	71	13	11	8	15	13
10	1604e24e	3587	**	**	**	**	0	1	3	2	**	**	**		1	52	64	9	71	13	11	8	15	13
90	1064a264	1936	88	**	..	86	88	**	86	..	..	..	..		**	..	..	..	..	..	..	..	..	..
91	1e6c4246	1926	52	52	61	61	**	**	..	..	52	61	52		..	**	..	..	..	..	..	..	..	..
92	1ab6f246	1895	53	53	64	64	85	85	**	**	53	64	53		..	85	**	..	**	..	..	..	..	..
93	1064f266	1888	11	11	9	9	18	18	19	19	11	**	11		..	18	19	**	19	**	**	89	**	..
94	1ec65246	1887	66	66	71	71	..	..	**	**	66	71	66		..	..	**	..	**	..	..	..	..	..
95	1604f266	1875	10	10	13	13	20	20	23	23	10	13	**		..	20	23	**	23	**	..	76	**	**
96	1064e246	1873	41	39	44	46	41	39	46	44	..	..	..		39	..	..	..	..	99	**	..	..	**
97	1a46e266	1872	8	8	15	15	24	24	29	29	**	15	8		..	24	29	98	29	98	..	**	**	..
98	1a46f266	1866	8	8	15	15	24	24	29	29	**	15	8		..	24	29	**	29	**	..	**	**	..
99	1604f24c	1849	41	39	44	46	41	39	46	44	..	..	..		39	..	..	..	..	**	**	..	..	**

Abb. 4.15: *Reaktionsmatrix zur Abbildung 4.13. Es handelt sich um einen Ausschnitt der Reaktionsmatrix der 100 häufigsten Spezies in der Generation 998. Das Symbol "**" bezeichnet eine elastische Wechselwirkung aufgrund des Replikationsverbots. Das Symbol ".." bedeutet, daß ein Reaktionsprodukt entsteht, dieses aber nicht unter den 100 häufigsten Spezies der Generation 998 zu finden ist.*
$M = 10^6$, Replikation verboten, Kode III, Run C1.

Crossover-Methode zur Umgehung des Replikationsverbots

Unter der Abwesenheit der NOT-Operation hat sich eine neue Methode zur Umgehung des Replikationverbots entwickelt. Sie ist dadurch gekennzeichnet, daß Reaktionsprodukte aus Teilsequenzen der Reaktionspartner zusammengesetzt werden (Abb. 4.16). Die entsprechenden Spezies sind in der Lage, ein Stück ihrer eigenen Binärsequenz in den Kode einer anderen einzupflanzen. Dies wird deutlich, wenn man sie mit den Spezies `00000000` und `ffffffff` reagieren läßt:

Quelle der Spezies (vgl. Abb. 4.16)	Spezies s	s + `00000000`	s + `ffffffff`
häufigste Spezies der Gen. 998	`1e64a24e`	`0000024e`	`fffff24e`
5. häufigste Spezies der Gen. 998	`1e64a246`	`0000024e`	`fffff24e`
176. häufigste Spezies der Gen 998	`1064f204`	`00000204`	`7ffff204`
200. häufigste Spezies der Gen. 998	`1e6c524c`	`1e6c4000`	`1e6c5ffe`

Abstrakte Beschreibung mittels einer algebraischen Struktur

Ein Simulationslauf produziert eine riesige Datenflut. Will man die 200 häufigsten Spezies über 2100 Generationen verfolgen, so fallen ca. 4 MB Daten an. In Diagrammen können

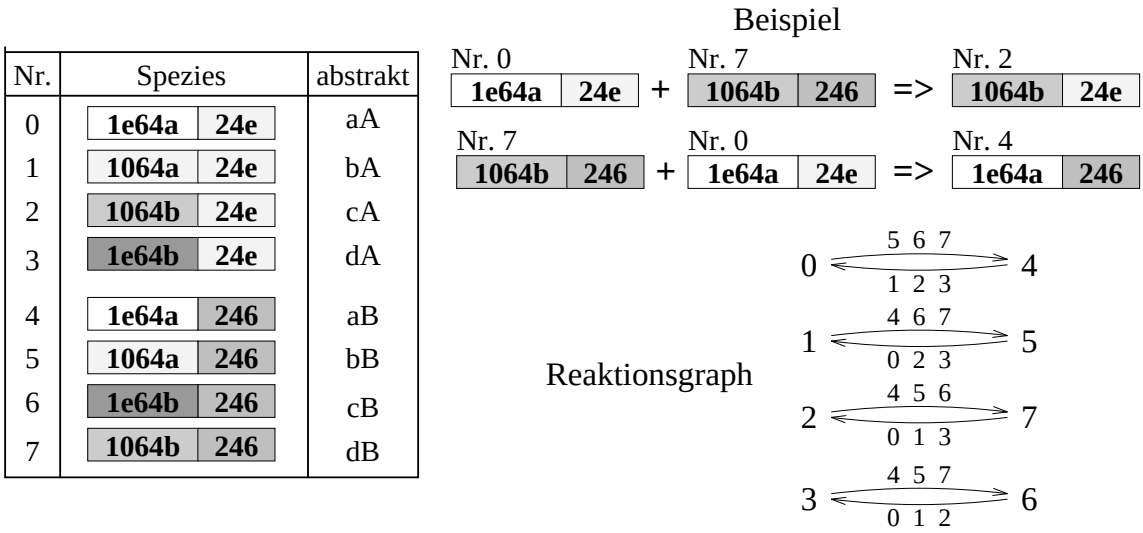


Abb. 4.16: *Crossover-Methode zur Umgehung des Replikationsverbots. Der Operator kopiert nur einen Teil seiner eigenen Binärsequenz in das Operandenregister, so daß das Produkt aus Teilsequenzen der Reaktanden zusammengesetzt ist. In dem Reaktionsgraph sind zur Verbesserung der Übersicht die Operatoren über den jeweiligen Reaktionspfeil geschrieben. Man erkennt, daß die Spezies eng gekoppelt sind (Run C1, vgl. Abb. 4.15).*

nur Bruchteile der enthaltenen Informationen dargestellt werden, indem man sie zu makroskopischen Größen – wie der Diversität und Produktivität – zusammenfaßt oder typische Details – wie den Konzentrationsverlauf einer Spezies – herausgreift.

Ähnliche Schwierigkeiten treten bei der statischen Beschreibung der Organisationsstruktur einer Population auf. Die hier verwendeten Reaktionsmatrizen und Reaktionsgraphen lassen zwar Regelmäßigkeiten erkennen, zeigen aber nur kleine Ausschnitte und werden schnell unübersichtlich und damit unhandlich.

Eine Alternative bietet sich in einer algebraischen Beschreibung an, die auch von FONTANA eingesetzt wird (vgl. Abschnitt 2.3.2). Dabei abstrahiert man völlig von dem zugrundeliegenden Reaktionsmechanismus. Im folgenden wird dies für die Organisation der 8 häufigsten Spezies in der Gen. 998 vorgeführt (vgl. Abb. 4.16 und 4.15).

Die Algebra bestehe aus der Menge M und dem Operator $\oplus : M \times M \to M$. M wird definiert als:

$$\begin{aligned} M &= X \times Y \qquad \text{mit} \\ X &= \{a, b, c, d\}, \\ Y &= \{A, B\} \end{aligned}$$

Jeder Spezies ist genau ein Element dieser Menge, entsprechend Abbildung 4.16, zugeordnet. Die Wechselwirkung zweier Spezies xy und $x'y'$ mit $x, x' \in X$ und $y, y' \in Y$ kann dann kompakt formuliert werden:

$$xy \oplus x'y' \longrightarrow x'y$$

Die gesamte, mindestens 6000 Spezies umfassende Organisation läßt sich nicht so einfach beschreiben, wie die beiden letzten Zeilen der vorhergehenden Tabelle andeuten. Vermutlich können aber automatische Verfahren zur Generierung von Algebren eingesetzt werden, um so Organisationen sichtbar zu machen, die nicht durch hohe Konzentrationen einzelner Spezies auffallen.

4.3.2 Oszillierendes Verhalten im Run C2 (Kode II, $M = 10^6$)

Zum Schluß soll ein Lauf vorgestellt werden, bei dem ein stark oszillierendes Verhalten zu beobachten ist. Er ist mit dem Kode II durchgeführt worden, der NOT durch EQ ersetzt. Im Vergleich zu Simulationen mit dem Kode III (NOT durch ID ersetzt) ist das Verhalten tendenziell komplexer, die Diversität höher und die Produktivität geringer (vgl. Abb. 4.17 und 4.13). Die Oszillation ist allerdings kein typischer Fall.

Die evolutive Phase ist noch ausgeprägter als in dem vorhergehenden Experiment (vgl. Abb. 4.18 und 4.14). Es sind unterschiedliche Entwicklungsschritte zu erkennen, die durch das Auftauchen einer Gruppe von Spezies in hoher Konzentration und das baldige – oft vollständige – Verschwinden derselben gekennzeichnet sind. In der Abbildung 4.18 ist für einige dieser Gruppen die jeweils häufigste Spezies dargestellt.

4.4 Diskussion

Die Experimente haben gezeigt, daß ein algorithmischer Reaktor mit der Automaten-Reaktion zur Selbstorganisation fähig ist. Nach einer zufälligen Initialisierung des Reaktors bildet sich selbständig eine nicht-triviale Organisationsstruktur heraus, ohne daß äußere Kräfte einwirken. Die Binärstrings treten als Organisatoren auf, wenn sie in ihrer Operatorform – dem abstrakten Automaten – andere Strings verarbeiten und neue produzieren. Der Entstehungsprozeß einer stabilen Organisationsstruktur ist – besonders bei Versuchen der Reihe C – ebenfalls nicht-trivial.

Das Replikationsverbot erwies sich als ein einfaches und effizientes Mittel, um elastische Kollisionen einzuführen und die Komplexität zu erhöhen. Es entstehen völlig neuartige, kooperative Spezies, die sich gegenseitig produzieren.

Die Versuchsreihe C hat gezeigt, daß die Verhaltensvielfalt durch die Entfernung einer "wichtigen" Operation (NOT) gesteigert werden kann. Die Spezies werden "herausgefordert", eine neue Methode selbständig zu entwickeln, um das Replikationsverbot ohne den Einsatz der NOT-Operation zu umgehen.

Bei der Vergrößerung der Population ist ein Phasenübergang im qualitativen Verhalten beobachtet worden. Für die Innovations- und Evolutionsfähigkeit ist eine Mindestgröße des Reaktors notwendig. Um diese ungenaue qualitative Aussage zu präzisieren, müssen die Innovations- und Evolutionsfähigkeit quantifiziert werden, und geeignete Meßverfahren zum Einsatz kommen. Insbesondere dann, wenn man später die Evolution der Evolutionsfähigkeit [Altenberg 1994] erfassen möchte.

Der Evolutionsprozeß hat den Charakter einer präbiotischen Evolution (Abb. 4.19). Im Gegensatz zur Darwinschen Evolution, bei der sich unterschiedliche Arten von Spezies mit einem weit verzweigten Stammbaum entwickeln, hat sich hier bei keiner Simulation

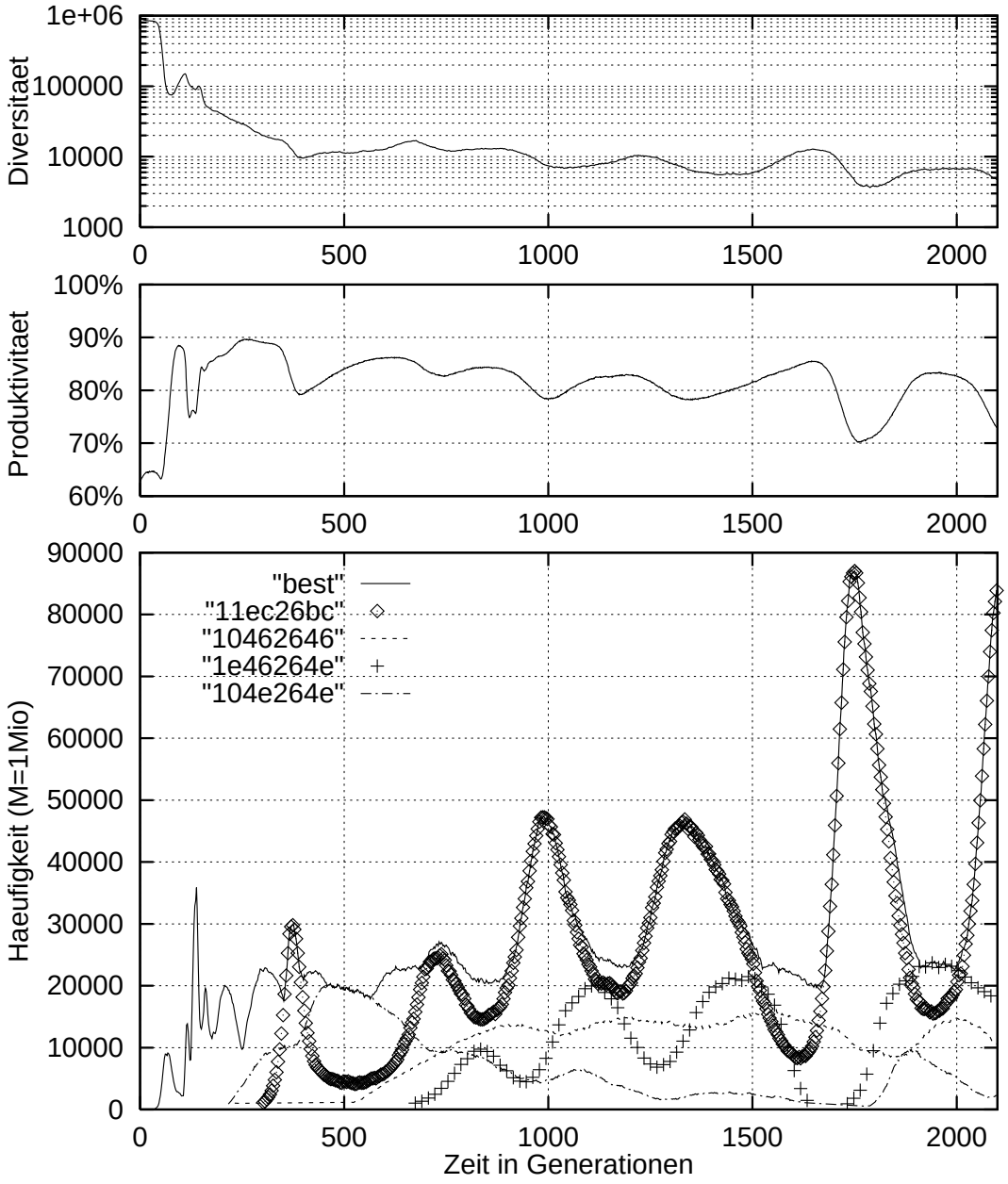


Abb. 4.17: *Oszillierendes Verhalten im Run C2. Hier ist im Gegensatz zum Run C1 die ID-Operation durch EQ ersetzt worden. Dies bewirkt im Mittel eine höhere Diversität, eine geringere Produktivität und ein komplexeres Verhalten. Die heftige Oszillation ist allerdings nicht typisch. Sie ist bei den 6 Versuchen mit derartig hohen Populationen nur einmal aufgetreten.*
$M = 10^6$, Replikation verboten, Kode II, Run C2.

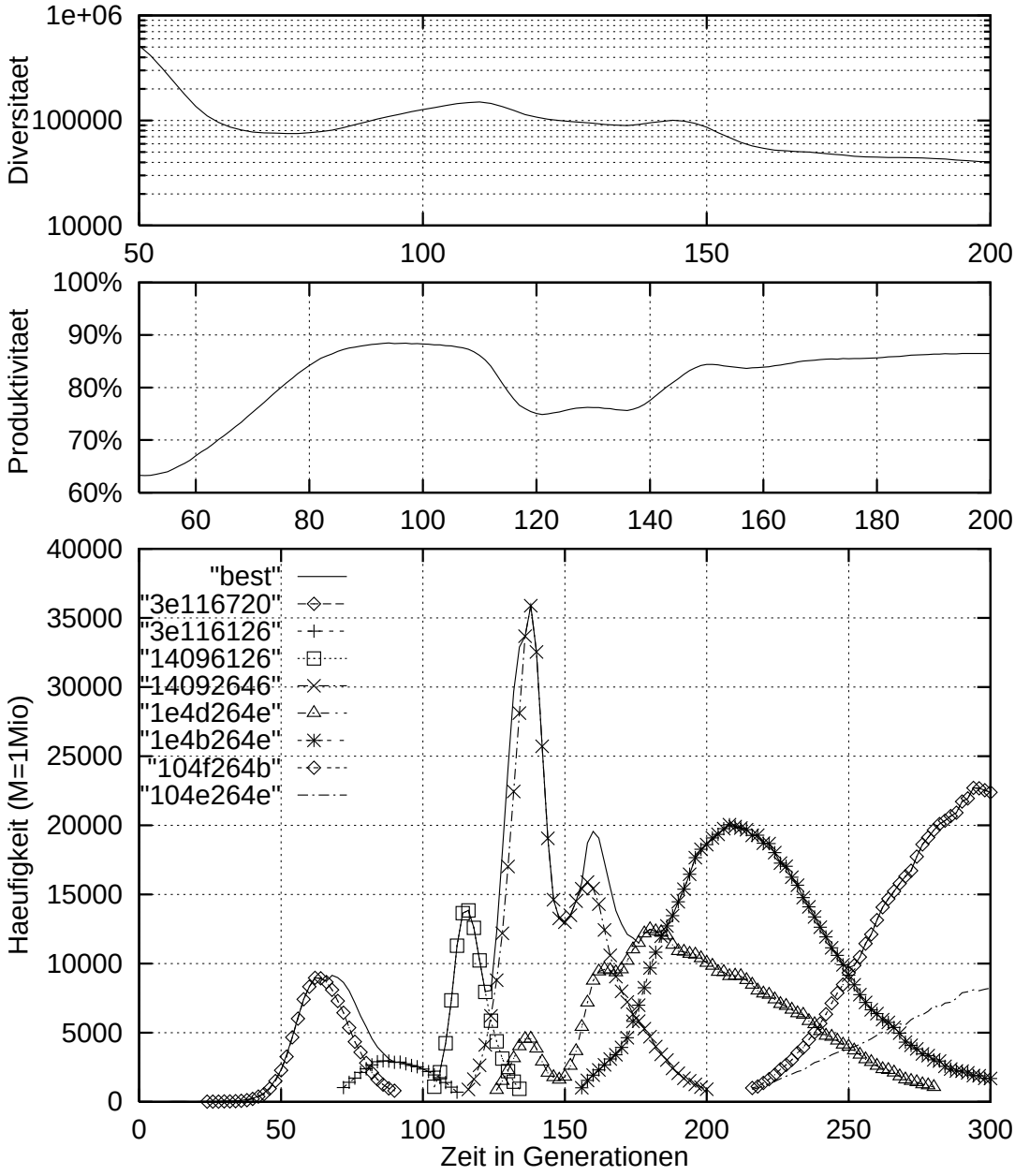


Abb. 4.18: *Ausschnittsvergrößerung der evolutiven Phase von Abbildung 4.17. Es sind deutlich unterschiedliche Entwicklungsschritte zu erkennen. Die dargestellten Spezies stehen stellvertretend für ganze Gruppen von Binärstrings.*
$M = 10^6$, Replikation verboten, Kode II, Run C2.

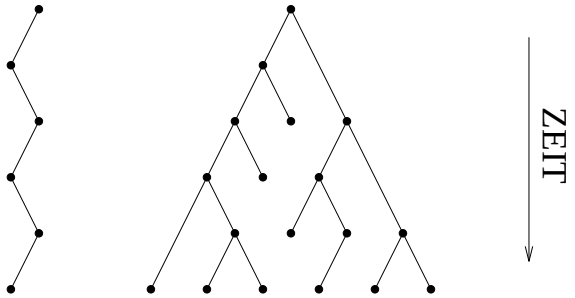


Abb. 4.19: *Abstammungsbäume der präbiotischen Evolution (links) und der Darwinschen Evolution (rechts) nach [*Hofbauer und Sigmund 1984*] Abb. 15.3.*

eine Organisation gebildet, in der Spezies mit grundsätzlich unterschiedlichem Aufbau stabil koexistieren konnten. Dieses Phänomen ist vermutlich auf die räumlich homogene Struktur der Population zurückzuführen. Es gibt keine Nischen, in denen sich verschiedene Speziesarten entwickeln können.

Das System ist zu einer "natürlichen" Evolution fähig, ohne daß externe, künstliche Operatoren für Replikation, Reproduktion, Mutation, Kreuzung oder Selektion, die bezüglich einer vorgegebenen Fitness erfolgt, eingesetzt werden. Allerdings findet eine implizite Selektion statt, die aufgrund der Kompetition um den beschränkten Reaktorraum entsteht.

Alle Variationen werden durch die Spezies selbst erzeugt. Damit liegt eine äußerst interessante Form von Hoftadters "seltsamen Schleife" vor [Hofstadter 1985]. Dadurch, daß die Spezies sich selbst verändern, verändern sie auch die Art und Weise, *wie* diese Modifikation erfolgt. Besonders deutlich ist dies bei der Entstehung des Crossover-Mechanismus in der Versuchsreihe C geworden, der eine Veränderung in der Evolutionsfähigkeit der Spezies bewirkt. Ein solches Verhalten kann man als Meta-Evolution bezeichnen. Da die Art der Meta-Evolution ebenfalls durch die Spezies bestimmt wird, ist auch eine Veränderung der Meta-Evolutionsfähigkeit denkbar und damit auch eine Meta-Meta-Evolution usw. Eine anscheinend endlose seltsame Schleife. Für Hofstadter ist sie ein wesentliches Element der (natürlichen) Intelligenz. Die Schleife entsteht hier durch die Operator-Operanden-Dualität der Spezies und den dadurch ermöglichten Selbstbezug.

Hat hier eine Meta-Evolution oder gar eine Meta-Evolution noch höherer Ordnung stattgefunden? Sicher ist, daß eine *Veränderung* der Evolutionsfähigkeit erfolgt ist. Für den Nachweis einer *Evolution* der Evolutionsfähigkeit müßten mehrere gerichtete Schritte ihrer Entwicklung aufgezeigt werden. Aufgrund der beschränkten Größe der Binärstrings und der beobachteten relativ raschen Reduzierung der Vielfalt ist eine Meta-Evolution höherer Ordnung äußerst unwahrscheinlich.

Der für die Reaktion definierte Automat besteht aus Elementen heute gängiger Computer. Er ließe sich leicht komponentenweise auf wenige Gatter einer realen elektronischen Schaltung abbilden. Es liegt deshalb die Vermutung nahe, daß nicht nur der hier verwendete Automat, sondern auch reale Computer die intrinsische Fähigkeit zur Selbstprogrammierung besitzen. Oder, auf Gatterebene ausgedrückt, es gibt elektronische Schaltungen mit einer "natürlichen" Fähigkeit zur Selbstverdrahtung.

Könnte man dieses Potential zur Evolvierung intelligenter Systeme nutzen, so ist ein erheblicher Effizienzgewinn zu erwarten, da aufwendige Abstraktionsebenen umgangen werden,

die durch Betriebssystemschichten, Programmiersprachen und Datenstrukturen gebildet werden. Offen ist allerdings, *wie* dies nutzbar gemacht werden kann. Denn eine Intelligenz, die sich in einem geschlossenen System, wie obigem algorithmischem Reaktor, entwickelt hat, wird völlig andersartig sein, weil sie nur Elemente ihrer Welt zu verstehen und zu manipulieren gelernt hat.

Kapitel 5

Informationsverarbeitung durch Reaktionssysteme

Ein Nutzungspotential von algorithmischen Reaktionssystemen ergibt sich – neben dem, daß sie als Studienobjekte für Organisationsphänomene dienen – aus ihrer Fähigkeit, Informationen zu verarbeiten bzw. Berechnungen durchzuführen. Damit ein System Informationen verarbeiten kann, muß es über die Möglichkeit einer Eingabe und einer Ausgabe von Daten verfügen. Es gibt drei Möglichkeiten, wie eine **Eingabe** erfolgen kann.

1. **Die Eingabe entspricht bestimmten Stoffmengen.**
   Die Eingabe erfolgt durch das Einbringen bestimmter Stoffe in den Reaktor. Die "Berechnung" kann *diskontinuierlich* erfolgen, indem der Reaktor zu Beginn einmal mit den Eingabestoffen und anderen benötigten Arbeitsstoffen befüllt und dann gestartet wird. Im *kontinuierlichen* Fall erfolgt ein ständiger Zufluß von Stoffen mit der Bedeutung einer Eingabe.

2. **Die Eingabe bestimmt den Selektionsdruck.**
   Dabei liegt die Eingabe als eine Gütefunktion $f : S \to \mathbb{R}$ vor. Ein Selektionsdruck kann erzeugt werden, indem bei jeder Auswahl eines Objekts aus der Population *zwei* gezogen werden und das bessere – entsprechend der Gütefunktion – verwendet wird.

3. **Die Eingabe bestimmt die Reaktionsbeziehungen.**
   Der Mechanismus zur Ermittlung des Reaktionsprodukts zweier Stoffe hängt dabei von der Eingabe ab und ist nicht wie in den bisher betrachteten Fällen fest. Zum Beispiel könnte bei einem System mit der Automaten-Faltung die Eingabe eine Kodierung der Befehle (vgl. Abb. 3.6) sein.

Kombinationen sind ebenfalls denkbar.

Die **Ausgabe** manifestiert sich in Konzentrationsanstiegen bestimmter Spezies. Zum Beispiel könnte der Stoff mit der höchsten Konzentration als Ausgabe interpretiert werden, oder – für ein Optimierungsproblem – das Objekt mit der höchsten Güte, unabhängig von seiner Häufigkeit.

5.1 Einführende Beispiele

5.1.1 Triviale Universalität

Es drängt sich zunächst die Frage auf, ob ein algorithmischer Reaktor universell im Sinne der Church-Turing-These ist. Oder anders ausgedrückt, ist er bezüglich der Berechenbarkeit genauso mächtig wie eine Turing-Maschine?

Es zeigt sich, daß diese Fragen präzisiert werden müssen. Man kann nämlich folgendermaßen leicht einen trivialen universellen Reaktor konstruieren, falls ein beliebiger Reaktionsmechanismus erlaubt ist.

Die Menge der Spezies S besteht aus den drei disjunkten abzählbaren Mengen der **Eingabespezies** S_I, der **Ausgabespezies** S_O und der **Arbeitsspezies** S_W. S braucht nur eine Arbeitsspezies zu enthalten, also $S_W = \{\mathbf{s}_w\}$. Die Funktion, die der Reaktor berechnen soll, ist $f : S_I \to S_O$.

Die Reaktionen zwischen zwei Elementen aus S seien wie folgt definiert:

$$\forall \mathbf{s}_i \in S_I, \mathbf{s}_o \in S_O \text{ mit } \mathbf{s}_o = f(\mathbf{s}_i): \quad \mathbf{s}_i + \mathbf{s}_w \Longrightarrow \mathbf{s}_o$$

Andere Reaktionen sind nicht erlaubt. Die entsprechende Reaktionsmatrix lautet also:

$$\mathrm{rm}(\mathbf{s}_i, \mathbf{s}_w) = \begin{cases} \mathbf{s}_o & \text{falls } \mathbf{s}_i \in S_I, \mathbf{s}_o \in S_O \text{ und } \mathbf{s}_o = f(\mathbf{s}_i), \\ elastic & \text{sonst} \end{cases}$$

Um den Reaktor zu benutzen, wird er halb mit $\mathbf{s}_w$ und halb mit einer Eingabe $\mathbf{s}_i$ befüllt. Nach einiger Zeit sind $\mathbf{s}_w$ und $\mathbf{s}_i$ größtenteils "verbraucht" (abgeflossen), und die gewünschte Ausgabe $\mathbf{s}_o = f(\mathbf{s}_i)$ liegt in hoher Konzentration vor.

Ein solcher Reaktor ist in dieser Form uninteressant, da die Rechenfähigkeit in der Definition der Reaktion steckt und nicht durch die Reaktordynamik bestimmt ist. Falls f eine beliebige durch eine Turing-Maschine berechenbare Funktion ist, muß zur Bestimmung des Reaktionsprodukts von $\mathbf{s}_i$ und $\mathbf{s}_w$ eine universelle Maschine verwendet werden. Mit anderen Worten, $\mathbf{s}_w$ – oder die Reaktionsmatrix, je nach Standpunkt – enthält die gesamte Information, wie zu einer Eingabe in endlicher Zeit eine Ausgabe zu berechnen ist, was den Reaktor überflüssig macht. Deshalb sind ausschließlich Reaktionssysteme von Interesse, bei denen die Interaktionen zwischen den Stoffen "einfach" sind, also von einem einfachen endlichen Automaten oder – aus technischer Sicht – von einer einfachen elektronischen Schaltung durchgeführt werden können.

Man kann durch Konstruktion zeigen, daß auch unter der Voraussetzung einer einfachen Interaktion zwischen den Stoffen ein Reaktor universell ist [Hjelmfelt, Weinberger, und Ross 1991]. Dabei muß man beliebig viele Spezies und einen beliebig großen Reaktor vorsehen, entsprechend dem beliebig langen Band einer Turing-Maschine. Die Berechnung erfolgt in mehreren Schritten, wie das folgende Beispiel zeigt.

5.1.2 Beispiel: Gerade Anzahl von Einsen

Ein Reaktor soll das Entscheidungsproblem lösen, ob die Anzahl der Einsen in einem Binärstring $\mathbf{s}_i$ gerade ist. Dazu wird der Reaktor mit n_i Einheiten der Eingabe $\mathbf{s}_i$ und n_w

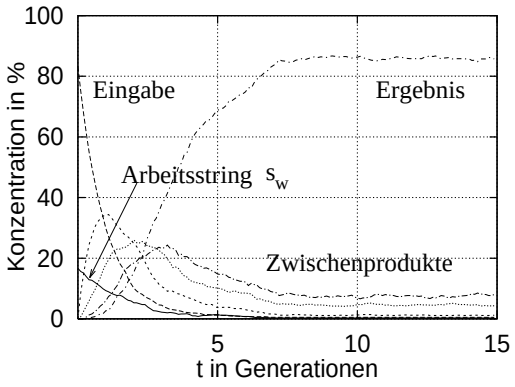


Abb. 5.1: *Simulation "gerade Anzahl von Einsen". Es werden nicht alle Zwischenprodukte umgesetzt. Die Menge von* $\mathbf{s}_w$ *reicht aber aus, um genügend Ergebnisstrings zu erzeugen. (*$M = 1000$, $n_i = 800$, $n_w = 200$, *6 verschiedene Spezies).*

Einheiten eines **Arbeitsstrings** $\mathbf{s}_w$ befüllt und gestartet. Die Ausgabe ist der Binärstring, der eine deutlich höhere Konzentration als alle anderen Stoffe hat. Handelt es sich bei dem String um $\mathbf{s}_0 = 0$, so bedeutet das, daß die Anzahl der Einsen des Eingabestrings gerade ist. Liegt der String $\mathbf{s}_1 = 1$ vor, so ist die Anzahl ungerade. Allgemein erfolgt die Reaktion von s_w als Operator mit einem anderen Binärstring $\mathbf{s}_i$ als Operand wie folgt:

- Ist der Operandenstring $\mathbf{s}_i$ gleich $\mathbf{s}_w$, $\mathbf{s}_0 = 0$ oder $\mathbf{s}_1 = 1$, so repliziere ihn. Diese Reaktion verstärkt ein Ergebnis ($\mathbf{s}_0$ oder $\mathbf{s}_1$), falls es auftaucht.
- Ist der Operandenstring gleich $\ldots b_2 b_1 b_0$, so erzeuge $\ldots b_2 b$ mit $b := (b_1 \neq b_0)$ und $b, b_0, b_1, b_2 \in \mathbb{B}$. Das heißt, daß die beiden letzten Bits des Operanden durch ihr EXOR-Produkt ersetzt werden und somit der String verkürzt wird. Dabei bleibt die Eigenschaft, ob der String eine gerade Anzahl von Einsen enthält, erhalten.

Für den Beispiellauf in Abbildung 5.1 mit der Eingabe $\mathbf{s}_i = \mathbf{s}_{01101} = 01101$ kommen folgende Reaktionsgleichungen zum tragen:

$$\begin{aligned}
\mathbf{s}_w + 01101 &\Longrightarrow 0111 \\
\mathbf{s}_w + 0111 &\Longrightarrow 010 \\
\mathbf{s}_w + 010 &\Longrightarrow 01 \\
\mathbf{s}_w + 01 &\Longrightarrow 1 \\
\mathbf{s}_w + 1 &\Longrightarrow 1 \qquad \text{(Replikation des Resultats)} \\
\mathbf{s}_w + \mathbf{s}_w &\Longrightarrow \mathbf{s}_w \qquad \text{(Selbstreplikation des Arbeitsstrings)}
\end{aligned}$$

Verwendet man den reinen Reaktor-Algorithmus 2.1.2, so sind die Arbeitsstrings nach einiger Zeit vollständig "verbraucht" (Abb. 5.1). Um dem entgegen zu wirken, kann eine ständige Zugabe von $\mathbf{s}_w$ erfolgen (Abb. 5.2).

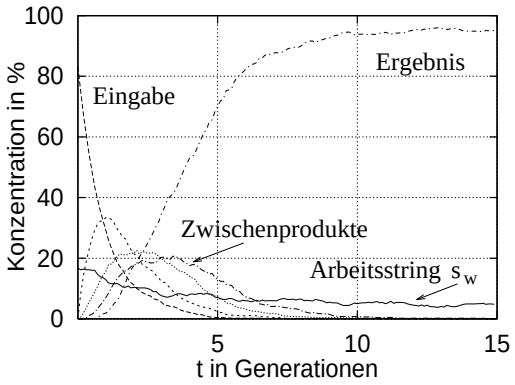


Abb. 5.2: *Konzentrationsverlauf "gerade Anzahl von Einsen" bei laufender Injektion der Arbeitsstrings s_w. Im Gegensatz zur Abbildung 5.1 werden auch alle Zwischenprodukte umgesetzt. ($M = 1000$, $n_i = 800$, $n_w = 200$, 6 verschiedene Spezies).*

5.1.3 Beispiel: Assoziativspeicher

Um die Konzentration mehrerer Arbeitsspezies zu erhalten und zu stabilisieren, müssen die Stoffe über Reaktionen miteinander zyklisch gekoppelt werden, falls man eine laufende externe Steuerung durch Zugabe diverser Spezies (wie in Abb. 5.2) vermeiden möchte ([Eigen und Schuster 1977], [Hofbauer und Sigmund 1984] S. 141-147). Im folgenden wird ein Reaktor gezeigt, der als Assoziativspeicher fungiert. Seine Daten werden in einem Hyperzyklus gespeichert, der von Arbeitsspezies gebildet wird.

Ein **Assoziativspeicher** ist ein Speicher mit wahlfreiem Zugriff. Der Zugriff erfolgt *nicht* auf eine durch eine Adresse spezifizierte Speicherzelle, sondern auf alle Speicherzellen gleichzeitig, die eine vorgegebene Inhaltsspezifikation erfüllen. Eine typische Anwendung ist der Cache-Speicher (vgl. [Engesser, Claus, und Schwill 1988] S. 38).

Dieses Beispiel definiert einen einfachen Speicher, der drei "Speicherzellen" enthält. Er kann zu drei verschiedenen Eingaben $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2$ die entsprechenden Ausgaben $\mathbf{s}'_0, \mathbf{s}'_1, \mathbf{s}'_2$ "assoziieren". Es werden drei Arbeitsstrings $\mathbf{s}_{w0}, \mathbf{s}_{w1}, \mathbf{s}_{w2}$ benötigt, die die einzelenen Zuordnungen realisieren. Im Ruhezustand enthält der Reaktor gleiche Anteile der drei Arbeitsstrings. Für einen Speicherzugriff werden nun Einheiten einer Eingabe hinzugegeben. Die entsprechende Ausgabe taucht als kurzer Konzentrationspeak auf. Danach geht der Reaktor wieder in einen Ruhezustand über, um weiter Anfragen bearbeiten zu können. Abbildung 5.3 zeigt einen Simulationslauf, bei dem zum Zeitpunkt 0 die Eingabe hinzugefügt wird.

Die Reaktionsgleichungen lauten:

$$\begin{aligned}
\mathbf{s}_0 + \mathbf{s}_{w0} &\Longrightarrow \mathbf{s}'_0 \\
\mathbf{s}_1 + \mathbf{s}_{w1} &\Longrightarrow \mathbf{s}'_1 \\
\mathbf{s}_2 + \mathbf{s}_{w2} &\Longrightarrow \mathbf{s}'_2 \\
\mathbf{s}_{w0} + \mathbf{s}_{w2} &\Longrightarrow \mathbf{s}_{w0} && (5.1) \\
\mathbf{s}_{w1} + \mathbf{s}_{w0} &\Longrightarrow \mathbf{s}_{w1} && (5.2) \\
\mathbf{s}_{w2} + \mathbf{s}_{w1} &\Longrightarrow \mathbf{s}_{w2} && (5.3)
\end{aligned}$$

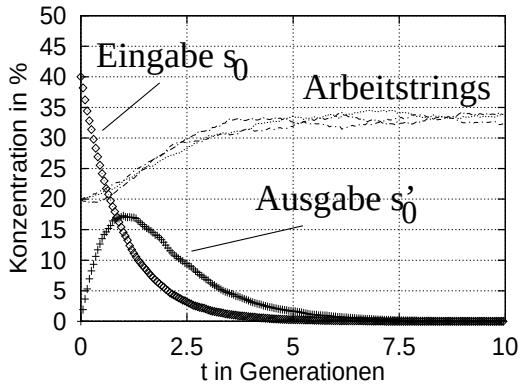


Abb. 5.3: *Simulation eines Speicherzugriffs. Zum Zeitpunkt Null werden 400 Objekte der Eingabe* $\mathbf{s}_0$ *eingefügt. Die gewünschte Ausgabe* $\mathbf{s}'_0$ *taucht als Konzentrationspeak auf und wird dann von den sich selbst produzierenden Arbeitsstrings verdrängt. Danach ist der Reaktor für weitere Zugriffe verfügbar. (*$M = 10000$*, 9 verschiedene Spezies).*

Durch die Gleichungen (5.1)-(5.3) werden die Konzentrationen der Arbeitsstrings stabilisiert. Sie bilden einen Hyperzyklus. Eine Definition der drei Gleichungen als Selbstreplikator-Gleichungen würde zu Instabilität der Konzentrationen der Arbeitsstrings führen (Abb. 5.4). Die Kopplung muß nicht unbedingt als Hyperzyklus erfolgen. Auch aus folgenden Reaktionsbeziehungen ergibt sich eine stabile Koexistenz der Arbeitsspezies:

$$\mathbf{s}_{w0} + \mathbf{s}_{w0} \Longrightarrow \mathbf{s}_{w1}$$
$$\mathbf{s}_{w1} + \mathbf{s}_{w1} \Longrightarrow \mathbf{s}_{w2}$$
$$\mathbf{s}_{w2} + \mathbf{s}_{w2} \Longrightarrow \mathbf{s}_{w0}$$

Ein derartiger Reaktionszyklus kann als Katalysator aufgefaßt werden, der Anfragen zu Anworten katalysiert. Abbildung 5.5 zeigt den für obige Simulation relevanten Reaktionsgraphen.

Der obige Speicher ist in dieser Form ein "read-only" Speicher. Um ihn beschreiben zu können, müßte man eine Reaktion definieren, die eine Eingabe mit der zugehörigen Ausgabe in den entsprechenden Arbeitsstring übersetzt. Der neue Arbeitsstring muß dann in den Hyperzyklus eingebunden werden.

5.2 Künstliche biochemische Reaktionssysteme

Es gibt mittlerweile eine Reihe von Arbeiten, die sich mit der Ausnutzung chemischer und biochemischer Prozesse zur Durchführung von Berechnungen beschäftigen. Man kann die verfolgten Ansätze in zwei Gruppen einteilen:

1. Die **analoge Methode**:
   Man versucht, bestehende komplexe Systeme, die Informationen verarbeiten (neuronale Netze, boolesche Netzwerke, Turing-Maschinen etc.), nachzubilden. Komponenten – wie zum Beispiel ein Gatter oder eine Verbindung – werden auf Teile des

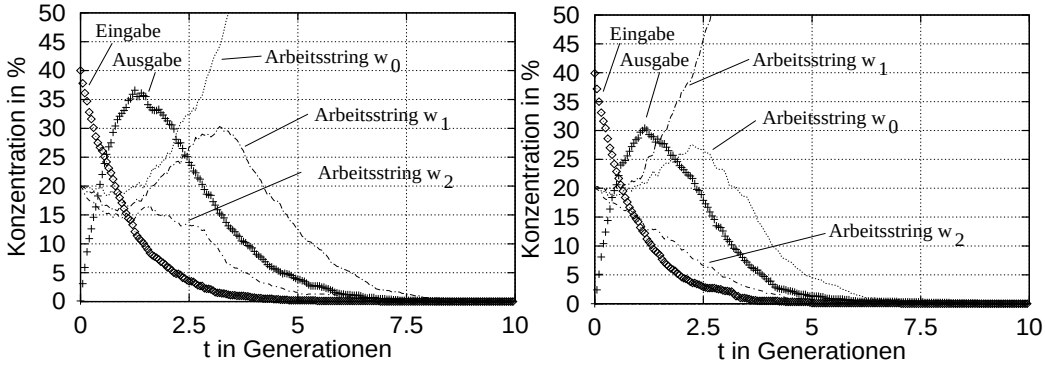


Abb. 5.4: *Simulation eines Speicherzugriffs. Im Gegensatz zur Abbildung 5.3 sind die Arbeitsspezies nicht zyklisch gekoppelt, sodern produzieren sich ausschließlich durch Selbstreplikationsreaktionen der Form* $\mathbf{s}_{wi} + \mathbf{s}_{wi} \Longrightarrow \mathbf{s}_{wi}$. *Dargestellt sind zwei typische Läufe mit unterschiedlichen Seeds.* ($M = 1000$, *9 verschiedene Spezies).*

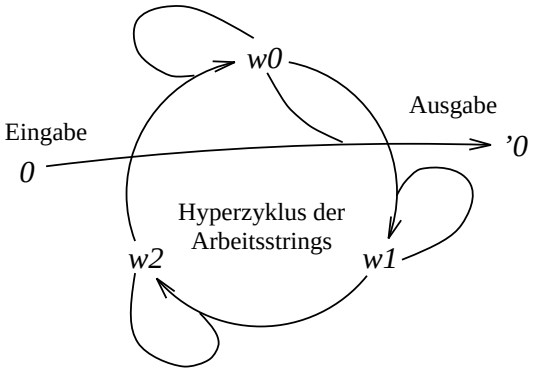


Abb. 5.5: *Auschnitt des Reaktionsgraphen, der für den Speicherzugriff der Abbildung 5.3 bedeutsam ist. Die Arbeitsspezies* $\mathbf{s}_{w0}$, $\mathbf{s}_{w1}$ *und* $\mathbf{s}_{w2}$ *bilden einen Hyperzyklus, der stabilisierend wirkt.*

chemischen Reaktionssystems abgebildet [Hjelmfelt, Weinberger, und Ross 1991; Hjelmfelt und Ross 1992; Arkin und Ross 1994].

2. Die **direkte Methode**:
   Es werden dabei die Eigenschaften des chemischen Systems möglichst direkt genutzt, um spezielle Aufgaben zu lösen (z.B. [Adleman 1994]).

5.2.1 Beispiel für die direkte Methode: Hamiltonsches Problem

Bei dem **Hamiltonschen Problem** geht es um die Frage: Hat ein gegebener Graph einen geschlossenen Weg, der alle Knoten genau einmal passiert? Das Entscheidungsproblem ist NP-vollständig und somit, aller Erfahrung nach, nicht in polynomialer Zeit zu lösen (vgl. [Engesser, Claus, und Schwill 1988] S.301). Eine ebenfalls NP-vollständige Variante des Problems lautet: Gibt es für einen gegebenen gerichteten Graphen einen Pfad, der mit dem Knoten v_{start} beginnt und mit v_{ende} endet, so daß jeder Knoten genau einmal besucht wird?

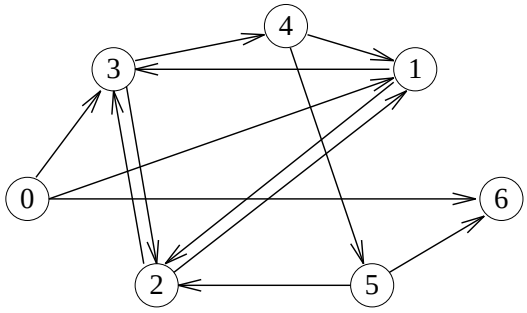


Abb. 5.6: *Gerichter Graph, der für $v_{start} = 0$ und $v_{ende} = 6$ einen eindeutigen Hamilton-Weg enthält:* $0 \to 1, 1 \to 2, 2 \to 3, 3 \to 4, 4 \to 5, 5 \to 6$. *(Nach [*Adleman 1994*] S. 1022).*

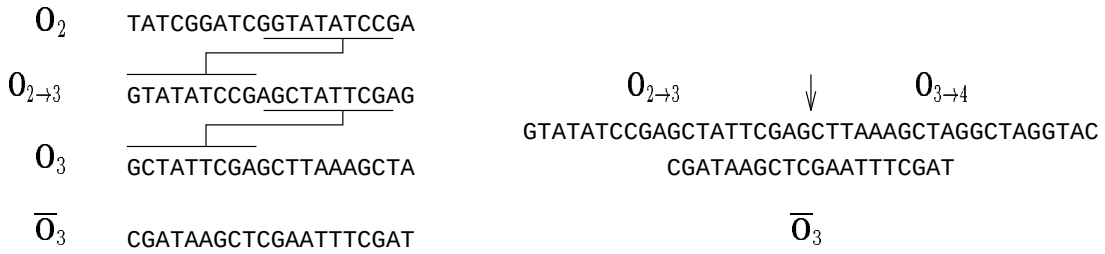


Abb. 5.7: *Links: Beispiele für Kodierung des Graphen als DNS-Sequenzen. Jedem Knoten i ist eine DNS-Sequenz $\mathbf{O}_i$ der Länge 20 zugeordnet. Eine Kante $i \to j$ entspricht einer DNS-Sequenz, die aus den letzten 10 Nukleotiden von $\mathbf{O}_i$ und den ersten 10 von $\mathbf{O}_j$ besteht. Rechts: Die Sequenz $\overline{\mathbf{O}}_3$ dient als Splint um $\mathbf{O}_{2\to3}$ mit $\mathbf{O}_{3\to4}$ zu verbinden. Dabei entsteht ein Molekül, das dem Pfad $2 \to 3, 3 \to 4$ entspricht. (Nach [*Adleman 1994*] S. 1022).*

Für diese Variante hat Leonard M. Adleman ein Verfahren vorgeschlagen, das mit den Werkzeugen der Molekularbiologie das Entscheidungsproblem löst [Adleman 1994]. Der zugrunde liegende Algorithmus erzeugt zunächst eine große Menge möglicher Pfade und löscht dann die Pfade, die nicht jeden Knoten genau einmal enthalten. Bleibt ein Pfad übrig, so ist die Entscheidungsfrage mit "ja" zu beantworten. Der Algorithmus hat wegen der vollständigen Enumeration aller möglichen Pfade exponentielle Laufzeit auf einer seriellen Maschine.

Für das Reaktionssystem werden kurze DNS-Sequenzen verwendet. Jedem Knoten i des Graphen ist eine eindeutige zufällige Sequenz $\mathbf{O}_i$ mit der Länge 20 zugeordnet: $\mathbf{O}_i = (O_{i,1}, \ldots, O_{i,20}) \in \{A, C, G, T\}^{20}$ (Abb. 5.7). Für das **Watson-Crick-Komplement** $\overline{\mathbf{O}}_i = (\overline{O}_{i,1}, \ldots, \overline{O}_{i,20})$ von $\mathbf{O}_i$ gilt:

$$\forall k \in 1, 2, \ldots, 20: \quad \overline{O}_{i,k} = \begin{cases} A & \text{falls } O_{i,k} = T, \\ T & \text{falls } O_{i,k} = A, \\ C & \text{falls } O_{i,k} = G, \\ G & \text{falls } O_{i,k} = C \end{cases}$$

Zu jeder gerichteten Kante $i \to j$ des Graphen gibt es eine ebenfalls 20 Nukleotide lange

DNS-Sequenz $\mathbf{O}_{i\to j} = (O_{i\to j,1}, \ldots, O_{i\to j,20})$. Sie besteht aus den letzten 10 Nukleotiden von $\mathbf{O}_i$ und den ersten 10 Nukleodiden von $\mathbf{O}_j$:

$$\forall k \in 1,2,\ldots,20: \quad O_{i\to j,k} = \begin{cases} O_{i,k+10} & \text{falls } k \leq 10, \\ O_{j,k-10} & \text{sonst} \end{cases}$$

Die einzelnen Schritte des Verfahrens werden nun erläutert, wie sie von ADLEMAN für den Graphen in Abbildung 5.6 durchgeführt worden sind:

1. **Erzeuge zufällige Pfade durch den Graphen.**
   Für jeden Knoten i des Graphen (außer 0 und 6) und jede Kante $i \to j$ werden 50 pmol von $\overline{\mathbf{O}}_i$ und 50 pmol von $\mathbf{O}_{i\to j}$ gemischt (1 pmol = Stoffmenge, die $6 * 10^{11}$ Teilchen enthält). Die $\overline{\mathbf{O}}_i$ wirken als Splinte, die die $\mathbf{O}_{i\to j}$ miteinander verbinden (Abb. 5.7, rechts). Es bilden sich also große Mengen von DNS-Molekülen, die gültige Pfade durch den Graphen repräsentieren.

2. **Behalte die Pfade, die mit $v_{start} = 0$ beginnen und mit $v_{ende} = 6$ enden.**
   Mit Hilfe der Polymerase-Kettenreaktion (engl. polymerase chain reaction, PCR, z.B. [VOET UND VOET 1992] S. 849) lassen sich DNS-Sequenzen verstärken. Die Verstärkung erfolgt selektiv nach Eigenschaften, die vorgegeben werden können. Für diesen Schritt werden die Sequenzen vervielfacht, die mit der Nukleotidsequenz des Knoten 0 beginnen und mit der des Knoten 6 enden. Innerhalb von 4 Stunden Laborarbeit kann eine Verstärkung um den Faktor 10^6 erreicht werden.

3. **Behalte davon die Pfade, die genau 7 Knoten enthalten.**
   Die Gelfiltrations-Chromatographie (z.B. [VOET UND VOET 1992] S. 84) wird eingesetzt, um Sequenzen auszufiltern, die 140 Basen lang sind.

4. **Behalte davon die Pfade, die jeden der 7 Knoten enthalten.**
   Dazu werden zunächst die Moleküle verstärkt, die die Sequenz $\mathbf{O}_1$ enthalten. Das heißt, daß die Pfade verbleiben, die wenigstens den Knoten 1 aufsuchen. Dieser Schritt wird für $\mathbf{O}_2$ bis $\mathbf{O}_5$ wiederholt.

5. **Bleibt ein Pfad übrig, so lautet die Antwort "ja", sonst "nein".**
   Die nach dem Schritt 4 verbliebenen Sequenzen werden wiederum verstärkt und sichtbar gemacht. Es ist auch möglich, die Struktur dieser Sequenzen zu bestimmen, um so nicht nur das Entscheidungsproblem zu lösen, sondern auch den entsprechenden Pfad zu erhalten (vgl. [ADLEMAN 1994] S. 1022).

Die beschriebene Technik erfordert ca. eine Woche Laborarbeitszeit, die durch Automatisierung verkürzt werden kann. Nach ADLEMAN sollte der Arbeits- und Zeitaufwand linear mit der Knotenanzahl anwachsen. Der Materialaufwand steigt exponentiell mit der Knotenanzahl.

Betrachtet man eine Anlagerungsreaktion, wie sie im 1. Schritt des Verfahrens stattfindet, als eine Operation, so müßte ein solches System laut ADLEMAN in der Lage sein, 10^{20} Operationen pro Sekunde durchzuführen, was um den Faktor 10 Mio. mehr ist, als heutige Supercomputer zu leisten vermögen. Dabei arbeitet das System äußerst energieeffizient. Es könnte ca. 10^{19} Operationen pro Joule ausführen. Im Gegensatz dazu kommen heutige Rechner auf "nur" 10^9 Operationen pro Joule ([ADLEMAN 1994] S. 1023).

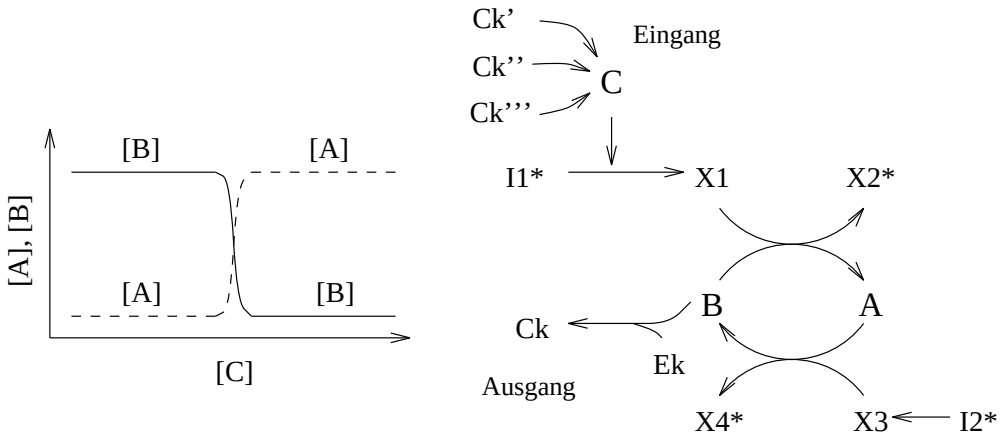


Abb. 5.8: *Schematische Struktur eines Reaktionsnetzwerks, das die Funktionalität eines McCulloch-Pitts-Neurons bzw. eines digitalen Gatters hat. Der linke Graph zeigt, wie der Zustand des Neurons von dem Eingangsstoff C beeinflußt wird. Stoffe, die mit einem "*" markiert sind, werden konstant gehalten (nach [*Hjelmfelt, Weinberger, und Ross 1991*] S. 10984 f.).*

5.2.2 Beispiel für die analoge Methode: Neuronales Netz

A. Hjelmfelt, E. D. Weinberger und J. Ross schlagen in [Hjelmfelt, Weinberger, und Ross 1991] eine Methode vor, wie ein neuronales Netz als chemisches Reaktionssystem implementiert werden kann. Das nachgebildete neuronale Netz besteht aus **McCulloch-Pitts-Neuronen**. Diese verfügen über eine Reihe von Eingängen, die mit skalaren Werten belegt werden können. Die Ausgabe ergibt sich als Anwendung einer Schwellenwertfunktion auf die gewichtete Summe der Eingangswerte [McCulloch und Pitts 1943].

Zur Realisierung eines Neurons wird ein reversibler Reaktionsmechanismus verwendet (Abb. 5.8). Er enthält die beiden Substanzen A und B, für die die Konservationsbedingung $[A]+[B] = const.$ gilt. Das bedeutet, eine Zunahme von A kann nur bei einer gleichzeitigen Abnahme von B erfolgen.

Der Katalysator C bestimmt das Konzentrationsverhältnis von A zu B. Eine hohe Konzentration von C bewirkt eine Produktion von $X1$, und dies wiederum eine Umsetzung von B zu A (Abb. 5.8). Dabei wird deutlich mehr B zu A umgesetzt, als umgekehrt durch die Reaktion $X3 + A \longrightarrow X4 + B$. Das Neuron befindet sich im **angeregten Zustand**, der durch eine hohe A Konzentration charakterisiert ist.

Falls $[C]$ einen kritischen Wert unterschreitet, so daß mehr A zu B als B zu A reagieren, erhält man eine hohe Konzentration von B und eine niedrige Konzentration von A, was dem **Ruhezustand** des Neurons entspricht (Abb. 5.8). Die Konzentration von C ist durch die Summe der "Eingangsstoffe" gegeben: $[C] = [Ck'] + [Ck''] + [Ck'''] + \ldots$.

Für den Aufbau eines Netzwerks benötigt man für jeden Knoten ein System obiger Art. Dabei müssen sich deren Substanzen chemisch unterscheiden, so daß Wechselwirkungen zwischen "internen" Stoffen unterschiedlicher Neuronen vermieden werden. Zur Verbindung der Neuronen wurde eine enzymatische Kopplung gewählt ([Hjelmfelt, Weinberger,

und Ross 1991] S. 10984). Dabei aktiviert A oder B ein Enzym Ek, das die Produktion von Ck bewirkt. Ck ist ein Eingangsstoff eines anderen oder desselben Neurons.

Soll ein Netzwerk in einem homogenen, gut gerührten Reaktionsgefäß untergebracht werden, so sind $4 + 4n + m$ unterschiedliche Substanzen von Nöten (n: Anzahl der Neuronen, m: Anzahl der Verbindungen, [Hjelmfelt, Weinberger, und Ross 1991] S. 10986). Die Anzahl der benötigten Stoffe kann reduziert werden, falls man eine räumliche Anordnung vorsieht. Ein Verbindung zwischen räumlich getrennten Reaktionssystemen kann durch Diffusion der Eingangsstoffe erfolgen.

Die wichtigsten Unterschiede zur Arbeit von Adleman sind in der folgenden Tabelle zusammengefaßt.

Adleman Hamiltonsches Problem	Hjelmfeld, Weinberger und Ross Neuronales Netz
Das Resultat ist ein einzelnes Molekül.	Die Ausgabe entspricht signifikanten Konzentrationsänderungen, für die eine große Anzahl gleicher Moleküle notwendig sind.
Ein elementarer Rechenschritt entspricht der reaktiven Kollision zweier Moleküle.	Es sind eine Vielzahl molekularer Interaktionen erforderlich, um in einem Neuron einen Konzentrationsanstieg – und somit eine elementare Operation – zu bewirken.
Es wird eine "mikroskopische" Eigenschaft des molekularen Systems direkt ausgenutzt, nämlich die komplementäre Anlagerungsreaktion.	Es wird die "makroskopische" Reaktionsdynamik ausgenutzt.
Das System ist empfindlich gegenüber geringen Verunreinigungen. Theoretisch genügt ein einzelnes Molekül, das eine unzulässige Kante kodiert, um das Ergebnis zu verfälschen.	Das System ist unempfindlicher gegenüber Verunreinigungen.
Der realisierte Algorithmus ist speziell und relativ einfach. Er löst natürlich ein schwieriges und häufiges Problem.	Das System ist universell, da mit Hilfe eines neuronalen Netzes jeder Algorithmus implementiert werden kann.

5.3 Metabolische Steuerung eines autonomen Roboters

In diesem Abschnitt wird anhand einer Simulation gezeigt, wie ein homogener Reaktor zur Steuerung eines autonomen mobilen Fahrzeugs bzw. Roboters eingesetzt werden kann [Dittrich und Banzhaf 1995].

5.3.1 Aufbau

Die simulierte **Welt**, in der sich der Roboter bewegt, besteht aus 400 **Feldern**, die auf einem quadratischen zwei-dimensionalen Gitter torusförmig angeordnet sind. In jedem Feld kann sich der Roboter und eine beliebige Anzahl der Substanzen A und B befinden.

Die Aufgabe des Roboters besteht darin, sich in der Welt zu bewegen und die Substanzen A und B aufzunehmen. Sie sollen dabei möglichst gleichmäßig absorbiert werden. Das heißt, daß nach der Aufnahme vieler Einheiten der Spezies A, die Bewegung in Richtung der Spezies B erfolgen soll, und umgekehrt.

Der Roboter besteht aus einem algorithmischen Reaktor der Größe M=4000, einem Motorsystem und je einem Sensor für die vier Himmelsrichtungen Westen (W), Norden (N), Osten (O) und Süden (S) (Abb. 5.9).

Über die **Sensoren** können Stoffe von den benachbarten Feldern aufgenommen werden. Bevor diese jedoch in den Reaktor wandern, werden sie von den Sensoren mit einer Markierung (engl. tag) versehen, die eindeutig für jeden Sensor ist. Zum Beispiel versieht der westliche Sensor die Stoffe mit der Markierung W, so daß ein durch ihn aufgenommenes A als AW in den Reaktor eingefügt wird. Spezies mit einer Markierung heißen **Sensorspezies**.

Im Reaktor werden im wesentlichen die Sensorspezies in die sogenannten **Motorspezies** W, N, O und S umgesetzt. Die Abbildung 5.10 zeigt das komplette Reaktionssystem des Reaktors.

Das **Motorsystem** ist in der Lage, den Roboter in eine der vier Himmelsrichtungen zu bewegen. Dazu mißt es (rückwirkungsfrei) die Konzentrationen der vier Motorspezies, indem in einer Probe von 500 zufällig gezogenen Spezies das Vorkommen von W, N, O und S gezählt wird. Falls eine eindeutige Majorität einer Motorspezies dabei festgestellt wird, erfolg eine Bewegung um ein Feld in die entsprechende Richtung.

Es folgt eine Übersicht der verwendeten Spezies:

- Die **Sensorspezies** AW, AN, AO, AS, BW, BN, BO und BS
  werden ausschließlich durch die Sensoren produziert, indem Einheiten von A und B von den Nachbarfeldern aufgenommen und mit einer Richtungsmarkierung versehen werden.
- Die **Motorspezies** W, N, O und S
  entstehen durch Reaktionen innerhalb des Reaktors.
- Die **Arbeitsspezies** A und B
  kommen sowohl in der Umwelt als auch im Reaktor vor. Sie werden direkt von dem Feld, auf dem sich der Roboter befindet, aufgenommen. Sie können *nicht* vom Roboter selbst produziert werden.
- Die **Abbauspezies** X
  wird durch diverse Reaktionen produziert, so daß sie einen Abbau der anderen Spezies bewirkt.

Die Simulation der Welt und des Roboters erfolgt durch den Algorithmus 5.3.1.

Algorithmus 5.3.1 *Simulation des metabolischen Roboters:*

- *Initialisierung*

  1. **Initialisierung der Welt.**
     Für die Simulation in der Abbildung 5.11 wird ein Feld, das mit einem A bzw. B gekennzeichnet ist, mit 700 Einheiten der jeweiligen Spezies befüllt.
  2. **Initialisierung des Reaktors.**
     Der Reaktor des Roboters wird (willkührlich) mit 1000 Einheiten A und 3000 Einheiten B befüllt.

- *Simulationsschritt*

  1. **Stoffaufnahme von den Nachbarfeldern durch die Sensoren.**
     Ein Drittel der As und Bs – maximal aber 150 Einheiten – werden von den vier Nachbarfeldern in den Reakor befördert, nachdem sie mit einer entsprechenden Richtungsmarkierung versehen worden sind.
  2. **Betrieb des Reaktors für 15 Generationen.**
     *Es erfolgen 15*4000 = 60000 Iterationen nach dem Reaktoralgorithmus 2.1.2.*
  3. **Aktivieren des Motorsystems.**
     Falls in einer Probe 500 zufällig gezogener Objekte unter den darin enthaltenen Motorspezies eine einfache Mehrheit vorliegt, wird der Roboter ein Feld in die Richtung der häufigeren Motorspezies bewegt. Die gezogenen Objekte verbleiben im Reaktor.
  4. **Absorbtion von A und B vom "eigenen" Feld.**
     Es werden alle Stoffe – maximal aber 1000 Einheiten – des aktuellen Feldes absorbiert und direkt in den Reaktor eingefügt.

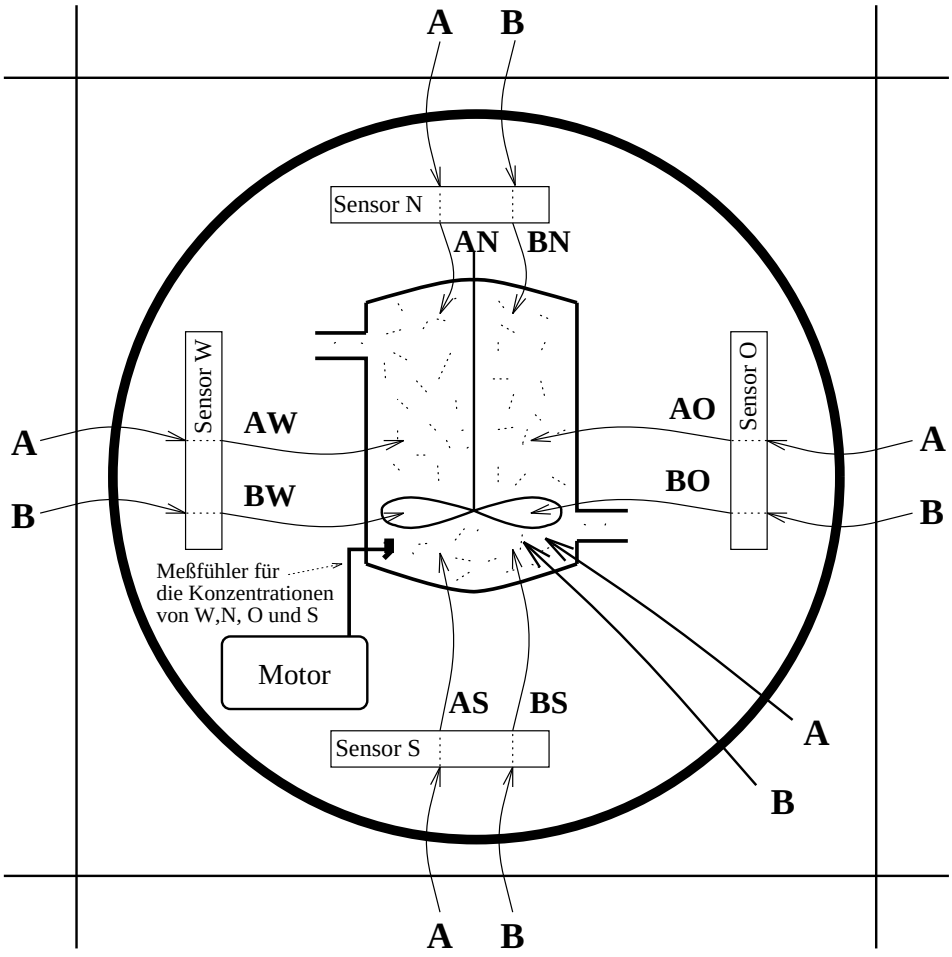
5.3.2 Simulation einer Pfadverfolgung

Die Abbildungen 5.11 zeigt die ersten 12 Simulationsschritte bei einer Pfadverfolgung. Der Pfad wird von den Substanzen A und B gebildet. Er ist so konstruiert, daß sich auf einem seiner Felder entweder 700 Einheiten von A oder 700 Einheiten von B befinden. Die Startposition des Roboters ist in der Abbildung 5.11 durch einen Kreis dargestellt.

Die Anordnung der Substanzen in der Form des obigen Pfades wurde gewählt, um das Verhalten des Systems besser nachvollziehen und darstellen zu können.

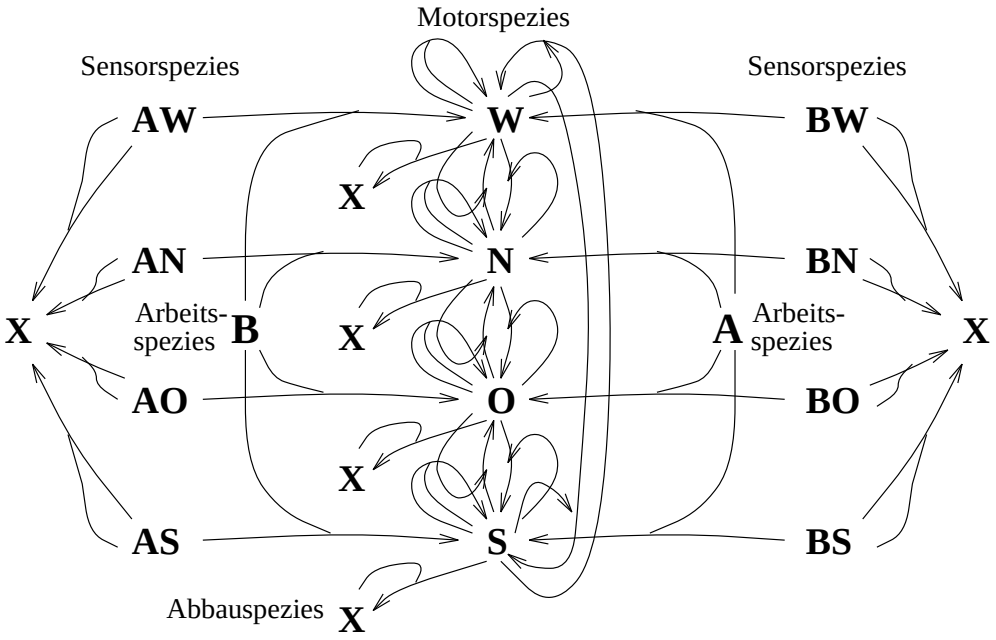
Der Pfad enthält eine Gabelung (Abb. 5.11). An der Gabelung sollte der Roboter laut Vorgabe sich nach rechts (Osten) bewegen, da er nach der vorausgehenden Aufnahme großer Mengen von A nun B aufnehmen muß.

Bei allen Experimenten mit unterschiedlichen Seeds für den Zufallszahlengenerator ist der Roboter dem Pfad exakt gefolgt. Die Abbildung 5.11 zeigt einen Ausschnitt aus einer typischen Simulation. Der Roboter bewegt sich zunächst nach Norden, da die durch den nördlichen Sensor eintretenden Stoffe NA und NB über die Reaktionen $B+AN \Longrightarrow N$ und $B+BN \Longrightarrow N$ eine hohe Konzentration der Motorspezies N bewirken. An der Gabelung herrscht im Reaktor ein Mangel an Arbeitsspezies B. Im Westen befindet sich eine hohe A



1.)

Abb. 5.9: *Schematische Struktur des metabolischen Roboters. Der Roboter ist auf einem Feld der Welt dargestellt. Ferner sind die Nachbarfelder mit ihren Substanzen A und B angedeutet. Diese können in geringen Mengen von den Sensoren aufgenommen werden. Die Sensoren versehen eingehende Substanzen mit einer Richtungsmarkierung. Vom aktuellen Feld können große Mengen von A und B direkt aufgenommen werden. Der Reaktor setzt im wesentlichen die eingehenden Stoffe in die Motorstoffe W, N, O, und S um. Der Motor kann nur die Konzentrationen der Motorstoffe messen und bewegt dementsprechend den Roboter.*



Spezies	AW	AN	AO	AS	BW	BN	BO	BS	A	B	W	N	O	S	X
AW	X									W					
AN		X								N					
AO			X							O					
AS				X						S					
BW					X				W						
BN						X			N						
BO							X		O						
BS								X	S						
A					W	N	O	S							
B	W	N	O	S											
W											W	N		S	X
N											W	N	O		X
O												N	O	S	X
S											W		O	S	X
X											X	X	X	X	

Abb. 5.10: Reaktionssystem des metabolischen Roboters. Wesentlich ist hier, daß die Sensorspezies AW, AN, AO, AS durch B und die Sensorspezies BW, BN, BO, BS durch A in die entsprechenden Motorspezies umgesetzt werden. Das bewirkt zum Beispiel, daß bei einer hohen A Konzentration und geringer B Konzentration hauptsächlich die Sensorspezies BW, BN, BO, BS umgesetzt werden. Dadurch erhöht sich die Wahrscheinlichkeit, daß der Motor den Roboter in die Richtung der benötigten Bs bewegt.

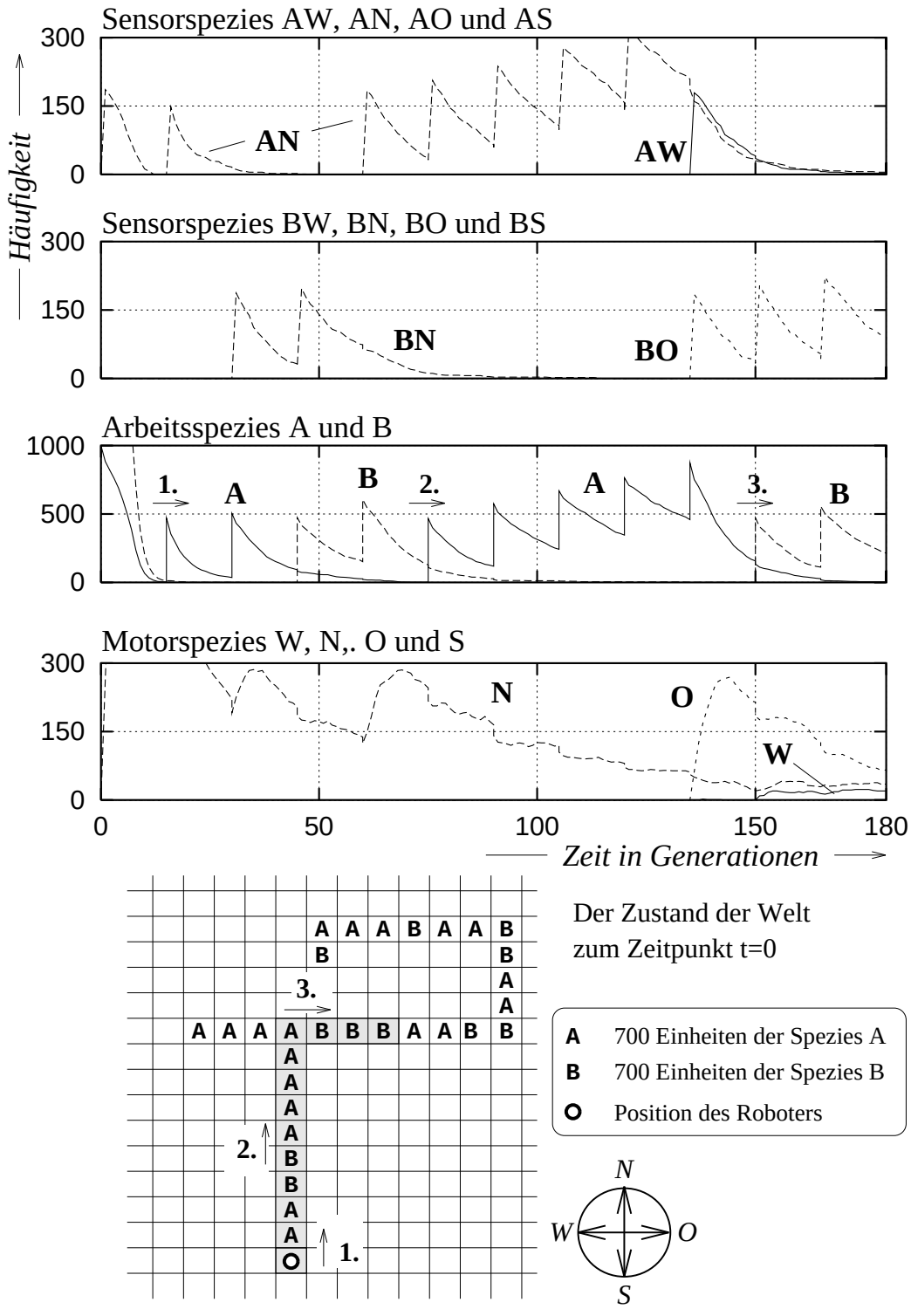


Abb. 5.11: *Verhalten des metabolischen Roboters an einer Pfadgabelung. Die obigen Graphen zeigen den Reaktorzustand über die Zeit für den grau unterlegten Teil des Pfades. An der Gabelung werden die Sensorstoffe AW und BO mit gleicher Häufigkeit aufgenommen. Da ein Mangel an B herrscht, und A im Überfluß vorhanden ist, werden hauptsächlich die BO von A in den Motorstoff O umgesetzt. Dagegen entstehen nur geringe Mengen W. Die Spezies O dominiert somit unter den Motorspezies, so daß der Roboter nach Osten bewegt wird.*

Konzentration und im Osten eine hohe B Konzentration, so daß die Sensorstoffe AW und BO in den Reaktor eindringen. Da die Konzentration der Arbeitsspezies A wesentlich höher als die von B ist, wird hauptsächlich die Motorspezies O erzeugt, gemäß $A + BO \Longrightarrow O$. Durch den Mangel an B entstehen über die Gleichung $B + AW \Longrightarrow W$ nur geringe Mengen der Spezies W. Dies führt zu der korrekten Entscheidung, den Roboter nach Osten zu bewegen.

Die Robustheit des Systems ist 1.) durch die Einbringung zufälliger Spezies in die Population und 2.) durch zufällige Modifikation der Reaktionsmatrix getestet worden.

Bezüglich zufälliger Fluktuationen in der Population erwies sich das System als äußerst stabil. Bei der Einbringung von bis zu 300 zufälligen Objekten pro Simulationsschritt ist der Roboter in der Lage, einem Pfad zu folgen. Vorausgesetzt, daß eine ausreichende Konzentration der Arbeitsspezies aufrecht erhalten werden kann. Bei mehr als 1000 zufälligen Objekten pro Simulationsschritt irrt er nur noch "ziellos" umher.

Verändert man die Reaktionsmatrix, so kann sich schon bei der Ersetzung eines einzigen Eintrags durch einen anderen Wert das Verhalten des Roboters drastisch ändern.

Ersetzt man zum Beispiel die Eintragung N in der Zeile B und der Spalte AN durch AW, so werden Sensorstoffe des nördlichen Sensors in die des westlichen umgesetzt. Der Roboter bewegt sich anstatt nach Norden nach Westen und ist nicht in der Lage, dem Pfad zu folgen.

5.3.3 Diskussion

Das Vorhergehende hat gezeigt, daß ein algorithmischer Reaktor in der Lage ist, ein System zu steuern. Die Wechselwirkung bzw. der Informationsaustausch mit der Umgebung wird dabei als Fluß von Substanzen und Berechnungen als Reaktionen zwischen den Substanzen aufgefaßt. Ein solches Konzept ist inhärent hochgradig parallel. Das Verhalten ergibt sich aus vielen einfachen Einzeloperationen, die keine globalen Informationen benötigen. Zur Reaktor-Motorsystem-Kopplung wird jedoch eine makroskopische Größe gemessen. Allerdings erfolgt durch das Ziehen einer Probe von 500 Objekten diese Messung (semi-) lokal. Es ist kein globaler Zugriff auf den gesamten Reaktor notwendig.

Es hat sich gezeigt, daß die Steuerung robust gegenüber Störungen im Populationsspeicher ist, nicht aber gegenüber strukturellen Störungen der Reaktionsbeziehungen. In einem realen technischen System würden die einzelnen Reaktionen parallel von vielen einfachen, gleichartigen Hardware-Einheiten ausgeführt werden, die möglicherweise sogar "fest verdrahtet" sind. Eine (globale) strukturelle Veränderung der Reaktionsbeziehungen ist dann unwahrscheinlich.

Die Empfindlichkeit des Systems gegenüber kleinen Änderungen in der Reaktionsmatrix macht seinen Entwurf "von Hand" äußerst schwierig. Die Empfindlichkeit resultiert insbesondere aus der Wettbewerbssituation, die durch die beschränkte Reaktorgröße entsteht. In der katalytischen Netzwerkgleichung drückt sich diese Eigenschaft im Verdünnungsterm

$$-x_k \Phi(t) = -x_k \sum_{i,j,k=1}^{n} \alpha_{ij}^k x_i x_j$$

aus (vgl. Abschnitt 2.2.2, Gleichung 2.5). Man sieht, daß das Wachstum jeder Spezies von den Konzentrationen aller anderen Spezies abhängig ist – bei nicht-trivialer Wahl der α_{ij}^k.

Zur Lösung der Entwurfsproblematik wären folgende Vorgehensweisen denkbar:

- **Entwurf durch ein externes "aufgesetztes" Optimierverfahren.**
  Dazu wird eine Fitnessfunktion definiert, die einem Reaktionssystem eine Güte zuordnet. Da es kaum möglich ist, diese in analytischer Form zu notieren, muß die Fitness experimentell bestimmt werden. Ist die Fitnessfunktion gegeben, so kann ein Standardverfahren, wie ein Evolutionärer Algorithmus, verwendet werden, um eine Reaktionsmatrix zu ermitteln.

- **Ausnutzung der Selbstorganisationsfähigkeit.**
  Die Experimente mit der Automaten-Reaktion haben gezeigt, daß ein algorithmisches Reaktionssystem zur Selbstkonstruktion und evolutiven Suche fähig ist, ohne daß dabei externe Operatoren zur Realisierung von Variation und Selektion zum Einsatz kommen. Um das System, das bisher nur seinen eigenen Interessen gefolgt ist, zur Lösung des gewünschten Problems zu bewegen, könnte zum Beipiel ein künstlicher Selektionsdruck eingesetzt werden.

- **Modularisierung durch Einführung einer räumlichen Struktur.**
  In der einfachsten Form würde dies eine Separation des Reaktors bedeuten, wobei ein "Sub-Reaktor" nur für eine kleine Teilmenge aller Spezies zuständig wäre. Zwischen den Sub-Reaktoren dürfen nur ausgezeichnete Spezies diffundieren, damit keine vollständige Vermischung möglich ist. Vorteilhaft ist ein solches Konzept, falls es Spezies gibt, die untereinander intensiv, mit anderen Spezies eher selten reagieren. Wird für eine solche Gruppe von Spezies ein eigener Reaktor eingerichtet, so erhöht sich auch die Effizienz, da die Anzahl elastischer Kollisionen reduziert wird.

Kapitel 6

Zusammenfassung und Ausblick

Im Mittelpunkt dieser Arbeit steht ein algorithmischer Reaktor, gefüllt mit Binärstrings von konstanter Länge 32. Bei einer Wechselwirkung zweier Strings wird der eine zu einem Automaten gefaltet, der den anderen String als Eingabe verarbeitet und einen neuen String als Reaktionsprodukt ausgibt.

Das Kapitel 4 zeigt, daß ein solches System zur Selbstorganisation, Innovation und Evolution fähig ist, ohne daß externe, künstliche Operatoren für Mutation, Rekombination oder Selektion notwendig sind. Die Spezies selbst treten als ihre eigenen Organisatoren auf. Aus vielen lokalen Einzelreaktionen haben sich komplexe Organisationsstrukturen kooperierender Spezies gebildet.

Betrachtet man den für die Reaktion verwendeten formalen Automaten als Stellvertreter realer Digitalrechner, so folgt die Vermutung, daß eine Computerarchitektur eine natürliche Eigenschaft zur Selbstoganisation bzw. Selbstprogrammierung hat, die mehr oder weniger ausgeprägt sein kann.

In Kapitel 5 wird gezeigt, daß ein Reaktor zur Verarbeitung oder Speicherung von Informationen eingesetzt werden kann. Für einen bestimmten Anwendungsfall sind dazu bisher die Reaktionssysteme "von Hand" konstruiert worden. Es ist naheliegend, hier die in Kapitel 4 beobachtete Fähigkeit zur Selbstorganisation anwendungsbezogen einzusetzen.

Die Anwendung ist in zwei unterschiedliche Szenarien denkbar:

1. Ein reales chemisches Systems wird zur Durchführung von Berechnungen eingesetzt. Dies lohnt sicher nicht für Probleme von geringer Komplexität, wie dem der "geraden Anzahl von Einsen". Auch ist vermutlich der Einsatz als universelle Workstation nicht sinnvoll. Vorstellbar ist aber die Verwendung für spezielle Aufgaben (z.B. Hammiltonsches Problem), zu deren Lösung ausgezeichnete Eigenschaften chemischer Systeme genutzt werden.

2. Das Paradigma eines chemischen Reaktors wird für die Programmierung bzw. Konstruktion von Parallelrechnern verwendet.

Was ist als Nächstes zu tun ?

- Das Intrumentarium zur Erkennung, Analyse und Visualisierung von Organisationsvorgängen ist zu verfeineren. Insbesondere gehört dazu 1.) ein praktisches Meßverfahren für die Innovativität und Evolutionsfähigkeit 2.) eine Methode, Organisationen zu erkennen, die nicht Spezies mit auffällig hohen Konzentrationen enthalten.
- Um möglicherweise eine Artenvielfalt zu erhalten, ist mit unterschiedlichen (räumlichen) Strukturierungen der Population zu experimentieren. Dabei sollte das Medium "Computer" respektiert werden. Eine zwei- oder drei-dimensionale räumliche Anordnung widerspricht dieser Forderung.
- Es sind Binärstrings variabler Länge zuzulassen. Erst dann ist eine Evolution mit offenem Ende prinzipiell möglich.

Literaturverzeichnis

Adleman, L. M. (1994, November). Molecular Computation of Solutions to Combinatorial Problems. *Science 266*, 1021–1024.

Altenberg, L. (1994). The Evolution of Evolvability in Genetic Programming. In K. E. Kinnear (Ed.), *Advances in Genetic Programming*, S. 47–74. Cambridge(MA)-London: A Bradford Book, The MIT Press.

an der Heiden, U. (1992). Selbstorganisation in dynamischen Systemen. In Krohn und Küppers (1992), S. 57–88.

Arkin, A. und J. Ross (1994, August). Computational Functions in Biochemical Reaction Networks. *Biophysical Journal 67*, 560–578.

Atkins, P. W. (1988). *Physikalische Chemie.* VCH, Weinheim-Basel-Cambridge-New York.

Banzhaf, M. und F. H. Eeckman (Eds.) (1995). *Evolution and Biocomputation: Computational Models of Evolution.* Springer, Berlin-Heidelberg-New York.

Banzhaf, W. (1990). The "molecular" traveling salesman. *Biological Cybernetics 64* , 7–14.

Banzhaf, W. (1993b). Self-replicating sequences of binary numbers. *Computers and Mathematics* (26), 1–8.

Banzhaf, W. (1993a). Self-replicating sequences of binary numbers – Foundations I and II: General and Strings of Length N = 4. *Biological Cybernetics* (69), 269–281.

Banzhaf, W. (1994). Self-organisation in a system of binary strings. In Brooks und Maes (1994).

Banzhaf, W. (1995). Self-Organizing Algorithms Derived from RNA Interaction. In Banzhaf und Eeckman (1995), S. 9–102.

Bethe, H. (1969). In *Les Prix Nobel en 1967*, S. 135. Stockholm.

Braitenberg, V. und I. Hosp (Eds.) (1994). *Entwicklung und Organisation in der Natur.* Rowohlt, Hamburg.

Brooks, R. und P. Maes (Eds.) (1994). *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA. MIT Press.

Church, A. (1941). The Calculi of Lambda-Conversion. In *Annals of Mathematical Studies*. no. 6. Princeton; Princeton University Press.

Cliff, D. (1991). Computational Neuroethology: A Provisional manifesto. In J.-A. Meyer und S. W. Wilson (Eds.), *From animals to animats*, S. 29–39. MIT Press.

Cliff, D., P. Husbands, J.-A. Meyer, und S. W. Wilson (Eds.) (1994). *From animals to animats 3: proceeding of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press, Bradford Book, Cambridge MA.

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Royal Society London A 400*(1818), 97 – 117.

Dittrich, P. und W. Banzhaf (1995). The Chemical Computation Metaphor and Robot Control. 3rd. European Conference on Artificial Life (ECAL 95), Granada, Posterbeitrag.

Eigen, M., W. Gardiner, P. Schuster, und R. Winkler-Oswatitsch (1981, Juni). Ursprung genetischer Information. *Spektrum der Wissenschaft* (6), 36–56.

Eigen, M. und P. Schuster (1977, November). The Hypercycle: a Priciple of Natural Self-Organisation, Part A. *Naturwissenschaften 64*(11), 541–565.

Eigen, M. und P. Schuster (1978a). The Hypercycle: a Priciple of Natural Self-Organisation, Part B. *Naturwissenschaften 65*, 7–41.

Eigen, M. und P. Schuster (1978b, Juli). The Hypercycle: a Priciple of Natural Self-Organisation, Part C. *Naturwissenschaften 65*(7), 341–369.

Eigen, M. und P. Schuster (1982). Stages of Emerging Life - Five Principles of Early Organisation. *J. Mol. Biol. 19*, 47–61.

Eller, B. (1995). Untersuchung von Selbstorganisationsphänomenen anhand von Nachbarschaftswechselwirkungen in einem Binärstringsystem. Diplomarbeit, Universität Dortmund, Fachbereich Informatik, Lehrstuhl für Systemanalyse.

Engesser, H., V. Claus, und A. Schwill (Eds.) (1988). *Duden INFORMATIK: Ein Sachlexikon für Studium und Praxis*. Mannheim-Wien-Zürich: Dudenverlag.

Farmer, J. D., S. A. Kauffman, und N. H. Packard (1986). Autocatalytic replication of polymers. *Physica D 22*, 50–67.

Feynman, R. P. (1986). Quantum mechanical computers. *Foundations of Physics 16*(6), 507 – 531.

Fontana, W. (1990). Algorithmic Chemistry: A New Approach to Functional Self-Organisation. Technical Report LA-UR 90-3431, Los Alamos National Laboratory.

Fontana, W. (1991). Algorithmic Chemistry. In Langton, Taylor, Farmer, und Rasmussen (1991), S. 159–209.

Fontana, W. (1994). Molekulare Semantik - Evolution zwischen Variation und Konstruktion. In Braitenberg und Hosp (1994), S. 69–106.

Fontana, W. und L. W. Buss (1994). The Arrival of the Fittest: Toward a Theory of Biological Organisation. *Bull. Math. Biol. 56*, 1–64.

Frauenfelder, H. (1988). Biomolecules. In D. Pines (Ed.), *Emerging synthesis of science*, SFI studies in sciences of complexity, vol. 1. Addison-Wesley, Reading.

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. *Monatshefte für Mathematik und Physik 38*, 173–198.

Grochia, E. (1978). *Einführung in die Organisationstheorie*. Stuttgart.

Haken, H. (1970). Laser Theory. In *Encyclopedia of Pysics, Vol. XXV/2c*. Springer, Berlin-Heidelberg-New York.

Haken, H. (1977). *Synergetics: An Indruduction*. Springer, Berlin-Heidelberg-New York-Tokyo.

Hillis, W. D. (1985). *The Connection Machine*. MIT Press, Cambridge, MA.

Hjelmfelt, A. und J. Ross (1992). Chemical implementation and thermodynamics of collective neural networks. *Proc. Natl. Acad. Sci. USA 89*, 388–391.

Hjelmfelt, A., E. D. Weinberger, und J. Ross (1991). Chemical implementation of neural networks and Turing machines. *Proc. Natl. Acad. Sci. USA 88*, 10983–10987.

Hofbauer, J. und K. Sigmund (1984). *Evolutionstheorie und dynamische Systeme*. Paul Parey, Berlin-Hamburg.

Höffgen, K. U., H. P. Siemon, und A. Ultsch (1991). Genetic Improvement of Feedforward Nets for Approximating Functions. In H.-P. Schwefel und R. M (Eds.), *Parallel Problem Solving from Nature — Proceedings 1st Workshop PPSN I*, Volume 496 of *Lecture Notes in Computer Science*, S. 302–306. Springer, Berlin.

Hofstadter, D. R. (1985). *Gödel, Escher, Bach:* (6. Auflage). Klett-Cotta, Originalausgabe: 1979, Basic Books, New York.

Jetschke, G. (1989). *Mathematik der Selbstorganisation - Qualtitative Theorie Deterministischer und Stochastischer Dynamischer Systeme*. Braunschweig: Vieweg.

Kauffman, S. A. (1990). Requirements for Evolvability in Complex Systems: Orderly Dynamics and Frozen Components. In W. H. Zurek (Ed.), *Complexity, Entropy and the Physics of Informatin,*, S. 151–192. Santa Fe Institute: Addison-Wesley.

Kauffman, S. A. (1993). *The Origins of Order*. Oxford University Press.

Krebs, H. (1964). In *Nobel Lectures, Physiology or Medicine 1942-1962*, S. 395. Amsterdam: Elsevier.

Krohn, W. und G. Küppers (Eds.) (1992). *Emergenz: die Entstehung von Ordnung, Organisation und Bedeutung* (1. Auflage). Frankfurt am Main: Suhrkamp.

Kruger, K. und et al. (1982). Self-splicing RNA: autoexcision and autocyclisation of the ribosomal RNA intervening sequence of Tetrahymena. *Cell 31*, 147–157.

Langton, C. G. (1989a). Artificial Life. In Langton (1989b), S. 1–45.

Langton, C. G. (Ed.) (1989b). *Artificial Life, SFI Studies in the Science of Complexity*. Santa Fe Institute, Los Alamos, New Mexico: Addison-Wesley.

Langton, C. G., C. Taylor, J. D. Farmer, und S. Rasmussen (Eds.) (1991). *Artificial Life II, SFI Studies in Science of Complexity, vol.X*. Santa Fe Institute, Los Alamos, New Mexico: Addison-Wesley.

Löffler, G. und P. E. Petrides (1984). *Physiologische Chemie* (4. Auflage). Springer, Berlin-Heidelberg-New York.

Maddox, J. (1985, August). Towards the quantum computer. *Nature 316*, 573.

May, R. M. (1991, Oktober). Hypercycles spring to life. *Nature 353*, 607–608.

McCulloch, W. S. und W. Pitts (1943). *Bull. Math. Biophys. 5*, 115–133.

Meyer, J.-A. und A. Guillot (1994). From SAB90 to SAB94: Four Years of Animat Research. In Cliff, Husbands, Meyer, und Wilson (1994), S. 2–11.

Miller, G. F., P. M. Todd, und S. U. Hedge (1989). Designing neural networks using genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, S. 379–384. Morgan Kaufmann Publishers, San Mateo, CA.

Miller, I. G. (1978). *Living Systems*. New York: Mac Graw Hill.

Morris, H. C. (1988). *Typogenetics: A Logic of Artificial Propagating Entities*. Ph. D. thesis, University of British Columbia, Vancouver, B.C.

Morris, H. C. (1989). Typogenetics: A Logic for Artificial Life. In Langton (1989b), S. 369–395.

Müller-Herold, U. (1992). Selbstordnungsvorgänge in der späten Präbiotik. In W. Krohn und G. Küppers (Eds.), *Emergenz: Die Entstehung von Ordnung, Organisation und Bedeutung*, S. 89–103. Suhrkamp.

Nussbaum, D. und A. Agarwal (1991, mar). Scalability of Parallel Machines. *Communications of the ACM 34*(3), 57–61.

Ray, T. S. (1991). An Approach to the Synthesis of Life. In Langton, Taylor, Farmer, und Rasmussen (1991), S. 371–408.

Schwegler, H. (1992). Systemtheorie als Weg zur Vereinheitlichung der Wissenschaften ? In Krohn und Küppers (1992), S. 27–56.

Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition. In Brooks und Maes (1994).

Stadler, P. F., W. Fontana, und J. H. Miller (1991, November). Random Catalytic Reaction Networks. (working paper of the SFI Series).

Sun, X.-H. und D. T. Rover (1994). Scalability of Parallel Algorithm-Machine Combination. *IEEE Transactions on Parallel and Distributed Systems 5*(6), 599–613.

Terzopoulos, D., X. Tu, und R. Grzeszczuk (1994). Artificial Fishes with Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World. In Brooks und Maes (1994).

Todd, S. und W. Latham (1991). Artificial Life or Surreal Art. In Varela und Bourgine (1991), S. 504–513.

Varela, F. J. und P. Bourgine (Eds.) (1991). *Towards a Practice of Autonomous Systems: Proceeding of the First Conference on Artificial Life.* MIT Press, Bradford Book, Cambridge.

Voet, D. und J. G. Voet (1992). *Biochemie.* Weinheim-New York-Basel-Cambridge: VCH.

von Neuman, J. (1966). *Theory of Self-reproducing Automata.* (ed. Arthur W. Burks), University of Illinois Press.

Welge, M. K. (1985). *Unternehmungsführung, Band 1: Planung.* Poeschel Verlag Stuttgart.

Welge, M. K. (1987). *Unternehmungsführung, Band 2: Organisation.* Poeschel Verlag Stuttgart.