TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik
Institut für Theoretische Informatik
Lehrstuhl für Automatentheorie

Splicing-P-Systeme

Bachelorarbeit

vorgelegt von: Benjamin Range
Matr.-Nr. 2809869

Betreuerin: Dr.-Ing. Monika Sturm

Hochschullehrer: Prof. Dr.-Ing. Franz Baader

eingereicht am: 11. August 2016

Inhaltsverzeichnis

1 Einleitung

Auf dem Gebiet des sogenannten *Natural Computing* finden sich verschiedene Zweige, die sich mit der Frage beschäftigen, in welcher Form Rechenvorgänge in der Natur stattfinden und wie Algorithmen oder gar ganze Rechnermodelle von der Natur inspiriert gestaltet werden können. Bekannte Beispiele hierfür sind genetische (oder allgemein evolutionäre) Algorithmen oder künstliche neuronale Netze. In dieser Arbeit beschäftigen wir uns mit zwei weiteren Teilgebieten, nämlich dem im Rahmen des molekularen Rechnens zu Bedeutung gelangten *DNA-Computing* und dem etwas später enstandenen sogenannten *Membrane Computing*.

Ersteres beschäftigt sich mit den Möglichkeiten, Berechnungen auf DNA vorzunehmen. Ein zentraler Punkt ist dabei die Tatsache, dass DNA-Stränge aufgrund ihrer mikrospisch kleinen Größe in sehr großer Anzahl vorliegen können, wobei Rechenoperationen auf diesen parallel ausgeführt werden. Mit den *Splicing-Systemen* steht ein mathematisches Modell zur Verfügung, welches einen biochemischen Vorgang beschreibt, der auch in der Praxis weitgehend kontrolliert werden kann. Wir werden hierzu sowohl einen Einblick in den biologischen Prozess des Splicings geben als auch das theoretische Modell näher vorstellen. Dabei werden wir auf eine für die Implementation eines auf diesem Modell beruhenden DNA-Rechners wichtige Beschränkung hinweisen, nämlich dass mit endlichen Ressourcen keine universelle Berechnungsstärke erreicht werden kann. Die entsprechenden Resultate entstammen [PRS98], wo bereits einige Ansätze diskutiert werden, wie diese Einschränkung theoretisch überwunden werden kann.

Im darauffolgenden Abschnitt betrachten wir sogenannte *Membran-* oder auch *P-Systeme*. Diese sind ein von Zellen und Membranen inspiriertes Modell, in welchem in einer hierarchisch angeordneten Struktur von Membranen auf darin befindliche Objekte Evolutionsregeln angewendet werden. Diese Anwendung erfolgt parallel auf allen Objekten einer Membran und für alle Membranen des Systems. Die Evolutionsregeln bieten dabei ein einfaches, aber mächtiges Beschreibungsmittel, mit dem sich z. B. auch die in einer Zelle ablaufenden chemischen Reaktionen und der Transport von Stoffen zwischen den Membranen modellieren lassen. Dies geschieht in einer stark abstrahierten Form, was einerseits den Vorteil mit sich bringt, komplexe Systeme nachvollziehbar modellieren zu können. Andererseits ist dadurch für eine Implementierung z. B. auf molekularer Ebene keine Grundlage gegeben. Im Gegensatz zu den Splicing-Systemen handelt es sich hierbei also um ein rein mathematisches Modell, welches auf absehbare Zeit nicht in Form eines tatsächlichen „Membranrechners“ realisiert werden kann. Eine Simulation auf traditioneller Rechentechnik ist möglich, büßt jedoch die parallele Arbeitsweise ein.

Der in Abschnitt 3 und 4 gegebene Überblick über Splicing- und Membransysteme bildet die Grundlage für den zentralen Teil dieser Arbeit. Dort widmen wir uns einer Kombination aus beiden Modellen, den *Splicing-P-Systemen*. Sie sind im Prinzip P-Systeme, bei denen in den Membranen anstatt der Evolutionsregeln das Splicing zum Einsatz kommt. Die Motivation, aus der die Zusammenführung erfolgt, setzt sich aus mehreren

Gesichtspunkten zusammen: Zum einen wird das Modell der P-Systeme auf diese Weise näher in den Bereich des technisch Umsetzbaren gerückt, da mit dem Splicing ein im Labor gut beherrschbarer Vorgang als Rechenoperation in den Membranen stattfindet. Von der anderen Seite betrachtet gilt es zu prüfen, ob eine Erweiterung der Splicing-Systeme um Membranen und die damit verbundenen Transportmöglichkeiten ein Mittel bieten, auch mit endlichen Ressourcen Turing-Mächtigkeit zu erlangen. Auch dieser Aspekt ist vor allem vor dem Hintergrund eines realisierbaren DNA-Rechners von Interesse. Inwieweit eine solche Entwicklung möglich sein wird, ist derzeit aber noch nicht abzusehen. Die hier vorgestellten Betrachtungen und Resultate beziehen sich auf das theoretische Modell und sind daher auch einfach im Kontext einer mathematischen Erforschung des Gebiets einzuordnen. Sie können somit aber einer Beantwortung der Frage dienen, welche Leistungsfähigkeit man sich von Splicing-P-Systemen erhoffen kann.

Dazu stellen wir auch eine häufig in Beweisen zur Betrachtung der Beschreibungsstärke angewandte Technik, das sogenannte *Rotate-and-Simulate* vor. Diese dient im Prinzip dazu, die Ableitungen einer formalen Grammatik durch ein Splicing-P-System zu simulieren, um daraus Aussagen über die Beschreibungsstärke des Systems abzuleiten.

Die hohe Parallelität der Berechnungsschritte in Splicing-P-Systemen erlaubt eine Behandlung auch sehr komplexer Probleme in attraktiver Zeitkomplexität. Wir werden dies abschließend am Beispiel eines NP-vollständigen Problems demonstrieren, indem wir ein Splicing-P-System entwerfen, welches dieses in polymomieller Zeit lösen kann.

2 Mathematische Vorbetrachtungen und Notationen

Seien M_1, M_2 beliebige Mengen. Wir schreiben $a \in M_1$, falls a *Element* der Menge M_1 ist. Die *Teilmengenbeziehung* notieren wir mit $M_1 \subseteq M_2$ bzw. für *echte* Teilmengen mit $M_1 \subset M_2$. Für *Vereinigung, Durchschnitt* und *kartesisches Produkt* der Mengen M_1 und M_2 schreiben wir $M_1 \cup M_2$, $M_1 \cap M_2$ bzw. $M_1 \times M_2$. Die *leere Menge* notieren wir mit $\emptyset$.

Die Menge der natürlichen Zahlen $\{0, 1, 2, \ldots\}$ bezeichnen wir mit $\mathbb{N}$, die Menge der positiven natürlichen Zahlen $\{1, 2, \ldots\}$ mit $\mathbb{N}^+$. Für beliebige $n \in \mathbb{N}$ definieren wir $[n] = \{m \in \mathbb{N} \mid m \leq n\}$ und $[n]^+ = \{m \in \mathbb{N}^+ \mid m \leq n\}$. Für beliebige $n \in \mathbb{N}^+$ und Mengen M definieren wir $M^1 = M$ und $M^{n+1} = M^n \times M$.

Ein *Alphabet* ist eine endliche, nicht-leere Menge V, deren Elemente wir *Symbole* nennen. Ein *Wort* über dem Alphabet V ist eine endliche Konkatenation von Symbolen aus V. Die *Länge* eines Wortes w notieren wir mit $|w|$. Das *leere Wort* der Länge 0 bezeichnen wir mit ε. Die n-fache Konkatenation eines Symbols $\alpha \in V$ bezeichnen wir mit α^n. Die Anzahl der Vorkommen eines Symbols $\alpha \in V$ in einem Wort $w \in V^*$ notieren wir mit $|w|_\alpha$. Die *Menge aller Wörter* über einem Alphabet V bezeichnen wir mit V^*, die Menge der *nicht-leeren* Wörter mit $V^+ = V^* \setminus \{\varepsilon\}$.

Eine *Sprache* über V ist eine Menge von Wörtern $L \subseteq V^*$. Seien V_1, V_2 Alphabete und $L_1 \subseteq V_1^*$, $L_2 \subseteq V_2^*$ Sprachen. Die *Konkatenation* der Sprachen L_1 und L_2 ist definiert als

$L_1 \circ L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. Zur Verkürzung der Notation schreiben wir hierfür auch nur $L_1 L_2$ und werden uns in diesem Zusammenhang zudem erlauben, einelementige Sprachen der Form $\{x\}$ nur durch das in ihnen enthaltene Element x zu repräsentieren. Aus dem Kontext wird dabei stets ersichtlich sein, ob es sich um Konkatenation von Symbolen oder Konkatenation von Sprachen handelt. $L_1 \# L_2$ steht demnach für den Ausdruck $L_1 \circ \{\#\} \circ L_2$. Für eine Sprache L definieren wir die *Längenmenge* von L als $ls(L) = \{|w| \mid w \in L\}$.

Eine *Grammatik* ist ein Tupel $G = (N, T, P, S)$, das folgende Bedingungen erfüllt: N ist ein Alphabet von *Nichtterminalsymbolen* und T ein Alphabet von *Terminalsymbolen*, wobei $N \cap T = \emptyset$. $S \in N$ ist das *Startsymbol* und P eine endliche Menge von *Produktionsregeln* mit $P \subset (N \cup T)^+ \times (N \cup T)^*$. Die Elemente von P notieren wir statt (β, γ) auch mit $\beta \to \gamma$.

Seien $u, u' \in (N \cup T)^*$. Wir definieren die Relation $u \Rightarrow_G u'$ falls $u = u_1 \beta u_2$, $u' = u_1 \gamma u_2$ und $\beta \to \gamma \in P$. Mit $\Rightarrow_G^*$ bezeichnen wir die reflexive und transitive Hülle von $\Rightarrow_G$. Die *von G erzeugte Sprache* ist $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$. Ein Wort $w \in (N \cup T)^*$ heißt *Satzform* von G, wenn gilt $S \Rightarrow_G^* w$.

Unter der *Chomsky-Hierarchie* verstehen wir die folgende Einteilung von Grammatiken (sowie der daraus erzeugten Sprachen): Jede Grammatik $G = (N, T, P, S)$ ist vom *Typ 0*, d. h. sie unterliegt keinen weiteren Beschränkungen. G ist *kontextsensitiv* bzw. vom *Typ 1*, falls für alle $\beta \to \gamma \in P$ gilt $|\beta| \leq |\gamma|$. Einzige erlaubte Ausnahme stellt die Regel $S \to \varepsilon$ dar, die die Ableitung des leeren Wortes ermöglicht. Falls $S \to \varepsilon \in P$, darf S auf keiner rechten Seite γ einer Regel $\beta \to \gamma \in P$ auftauchen. Wenn für alle Regeln $\beta \to \gamma \in P$ einer Typ-1-Grammatik zusätzlich $\beta \in N$ gilt, ist G *kontextfrei* bzw. vom *Typ 2*. Eine Typ-2-Grammatik G ist *regulär* oder vom *Typ 3*, wenn $\gamma \in T \cup TN$ gilt für alle $\beta \to \gamma \in P$.Eine Sprache ist vom Typ 0 (Typ 1, Typ 2, Typ 3), falls es eine Typ-0- (Typ-1, Typ-2, Typ-3)-Grammatik G gibt, so dass $L(G) = L$. Wir verwenden auch die Bezeichnungen rekursiv aufzählbare, kontextsensitive, kontextfreie und reguläre Sprachen. Eine Grammatik G heißt *linear*, falls für alle $\beta \to \gamma \in P$ gilt: $\gamma \in T^* \cup T^* N T^*$, *rechts-linear* mit $\gamma \in T^* \cup T^* N$ und *links-linear*, falls $\gamma \in T^* \cup N T^*$.

Die bis hier gegebene Beschreibung formaler Grammatiken folgt [SCH01], allerdings ist der Begriff Satzform dort etwas weiter gefasst und beinhaltet beliebige Wörter $w \in (N \cup T)^*$. Unter einer Satzform verstehen wir in dieser Arbeit (analog zu [PRS98] und [Pău02]) jedoch nur Wörter, die vom Startsymbol beginnend abgeleitet werden können.

Eine Typ-0-Grammatik ist in *Kuroda-Normalform*, wenn alle Produktionsregeln $p \in P$ von folgender Form sind: $AB \to CD$, $A \to CD$, $A \to a$ oder $A \to \varepsilon$, mit $A, B, C, D \in N$ und $a \in T$. Zu jeder Grammatik G existiert eine äquivalente Grammatik G' in Kuroda-Normalform, so dass $L(G) = L(G')$ (vgl. [Pău02], Theorem 2.3.2).

Die Klassen aller rekursiv aufzählbaren, kontextsensitiven, kontextfreien bzw. regulären Sprachen bezeichnen wir mit RE, CS, CF bzw. REG. Hierbei gilt (vgl. [SCH01]) $REG \subset CF \subset CS \subset RE$. Die Klasse aller endlichen Sprachen bezeichnen wir mit FIN. Für eine Klasse $\mathcal{K} \in \{RE, CS, CF, REG, FIN\}$ ist die Klasse $N\mathcal{K} = \{ls(L) \mid L \in \mathcal{K}\}$. Hierbei

sprechen wir z. B. für $NREG$ von der Klasse regulärer Mengen natürlicher Zahlen.

Für ein Alphabet $V = \{\alpha_1, \ldots, \alpha_n\}$ definieren wir die *Parikh-Abbildung* $\Psi_V \colon V^* \to \mathbb{N}^n$ mit $\Psi_V(w) = (|w|_{\alpha_1}, \ldots, |w|_{\alpha_n})$ für alle $w \in V^*$. Für eine Sprache $L \subseteq V^*$ bezeichnet das *Parikh-Bild* von L die Menge $\Psi_V(L) = \{\Psi_V(w) \mid w \in L\}$. Die Menge aller Parikh-Bilder der Sprachen einer Klasse $\mathcal{K} \in \{RE, CS, CF, REG, FIN\}$ notieren wir mit $Ps\mathcal{K}$ (z. B. bezeichnet $PsRE$ die Menge der Parikh-Bilder aller rekursiv aufzählbaren Sprachen).

In den Kapiteln 4, 5 und 6 werden wir Graphen als Darstellungsform verwenden, weswegen abschließend noch die folgenden Definitionen gegeben werden: Ein *gerichteter Graph* G ist ein Paar $G = (V, E)$, bestehend aus einer Menge V von *Knoten* und einer Menge $E \subseteq V \times V$ von *Kanten*. Ein *Pfad* von G ist eine Folge $v_1, \ldots, v_n$ mit $n \in \mathbb{N}^+$, $v_i \in V$ für $i \in [n]^+$ und $(v_j, v_{j+1}) \in E$ für $j \in [n-1]^+$. Die *Länge* eines solchen Pfades ist n. Ein *Baum* ist ein gerichteter Graph, bei dem ausgehend von einem *Wurzelknoten* $v \in V$ zu jedem Knoten $v' \in V$ genau ein Pfad existiert. Unter der *Tiefe* eines Baums B verstehen wir die maximale Länge aller Pfade von B (d. h. der Baum $B_1 = (\{v_1\}, \emptyset)$ hat die Tiefe 1).

3 Splicing

3.1 Biologischer Hintergrund

Die im DNA-Computing betrachteten Rechenmodelle arbeiten im Allgemeinen mit Zeichenfolgen über beliebigen Alphabeten. Sie stellen jedoch eine Abstraktion von biochemischen Vorgängen dar, die auf DNA-Strängen – sowohl in der Natur als auch vom Menschen initiiert – auftreten können. Daher ist es angebracht, zunächst einen Überblick über die zu Grunde liegenden biologischen Strukturen und Prozesse zu geben.

Nun sind die davon inspirierten Modelle auf mathematischer Ebene in vielfältiger Art und Weise modifiziert und weiterentwickelt worden – zumeist mit der Absicht, die daraus resultierenden Einflüsse auf die Berechnungsstärke des Modells zu untersuchen – und haben damit teilweise den Rahmen der im Labor umsetzbaren Möglichkeiten verlassen (z. B. durch Voraussetzung unendlich großer DNA-Mengen). Ein zentrales Ziel dieser Untersuchungen bleibt jedoch, mit endlichen Ressourcen einen möglichst leistungsfähigen „DNA-Rechner“ implementieren zu können, was eine zusätzliche Motivation darstellt, die im folgenden Abschnitt beleuchteten Ursprünge nicht aus den Augen zu verlieren.

Im Rahmen dieser Arbeit beschränken wir uns hierbei ähnlich zu [Fri08] auf die für das Verständnis der späteren Kapitel nötigen oder mindestens nützlichen Aspekte. Eine fundiertere, jedoch ebenfalls an Leser mit informatisch-mathematischem Hintergrund gerichtete Darstellung findet sich in [PRS98] (S. 9-41).

3.1.1 Die Struktur der DNA

Die *DNA* (vom englischen Begriff *deoxyribonucleic acid*, im Deutschen *Desoxyribonukleinsäure* und entsprechend mit *DNS* abgekürzt; in dieser Arbeit wird im Kontext der Quellenliteratur jedoch das englische Akronym verwendet) ist ein Kettenmolekül, das aus zwei, in der Art einer Doppelhelix umeinander gewundenen Strängen besteht. Jeder Einzelstrang wiederum ist eine Verknüpfung von speziellen Bausteinen, den *Nukleotiden*. Jedes dieser Nukleotide setzt sich aus drei Komponenten zusammen: dem Zucker *Desoxyribose*, einem *Phosphatrest* und einer *Nukleinbase*. Dabei treten in der DNA vier verschiedene Arten solcher Basen auf, nämlich *Adenin*, *Guanin*, *Cytosin* und *Thymin*, welche gemeinhin mit ihren Anfangsbuchstaben abgekürzt werden: `A`, `G`, `C` und `T`.

Abbildung 1 zeigt die chemische Struktur eines Nukleotids mit Nukleinbase Thymin. Die fünf Kohlenstoffatome des Zuckers werden hierbei durchnummeriert und mit $1'$ bis $5'$ bezeichnet.[1] Dabei ist $1'$ die Stelle, an welcher die Base anknüpft, wohingegen die Phosphatgruppe mit $5'$ verbunden ist. Diese etwas detailliertere Betrachtung der Desoxyribose ist dadurch motiviert, dass die Verbindungen innerhalb eines DNA-Einzelstrangs durch eine Phosphodiesterbindung der $5'$-Phosphatgruppe eines Nukleotids mit der am $3'$-Kohlenstoffatom des benachbarten Nukleotids verknüpften Hydroxygruppe zustande kommen. Dadurch lässt sich insbesondere mit Hinblick darauf, DNA-Stränge als Zeichenreihen zu repräsentieren, eine eindeutige Leserichtung festlegen. Folglich bezeichnet die Folge $5'$-`GAATTC`-$3'$ die Verbindung von Nukleotiden, bei der sich die freie Phosphatgruppe am $5'$-Ende von `G` befindet und eine ungebundene Hydroxygruppe am $3'$-Ende von `C`.

Zwischen den Basen zweier Nukleotide bilden sich sogenannte *Wasserstoffbrückenbindungen* aus, wobei dies nur innerhalb der Paarungen Adenin – Thymin sowie Guanin – Cytosin möglich ist. In einem DNA-Doppelstrang findet man somit zu jedem Nukleotid auf der gegenüberliegenden Seite den jeweils komplementären Baustein, also `A` gegenüber `T` und `G` gegenüber `C`. Dabei ist festzuhalten, dass die Leserichtung der beiden komplementären Stränge einander entgegengesetzt verläuft, sich also am $5'$-Ende des einen Stranges das $3'$-Ende des gegenüberliegenden befindet. Betrachten wir z. B. die Kette $5'$-`AATCAG`-$3'$, so wäre der zugehörige Komplementärstrang die Folge $3'$-`TTAGTC`-$5'$.

DNA-Moleküle können in vielfältiger Art und Weise vorliegen, z. B. überlappend, mit Unterbrechungen, Schlaufen oder gar ringförmig. Einer dieser Fälle ist für das später genauer betrachtete Splicing essenziell und soll daher an dieser Stelle kurz veranschaulicht werden:

Wenn an einem Ende eines Doppelstranges einer der beiden Einzelstränge einige Nukleotide weiter hinausragt, so spricht man von einem sogenannten „*sticky end*“, also einem „klebrigen Ende“. Die Bezeichnung rührt daher, dass ein weiterer Strang mit dazu passendem – also komplementären – „sticky end“ sich an dieser Stelle anheften kann. Alternativ dazu spricht man bei einem Ende ohne Überhang von einem „*blunt end*“. Abbildung 2

[1] Die Kennzeichnung mit „′“ dient dazu, Mehrdeutigkeiten zu vermeiden, da die Base ebenfalls Kohlenstoff enthält.

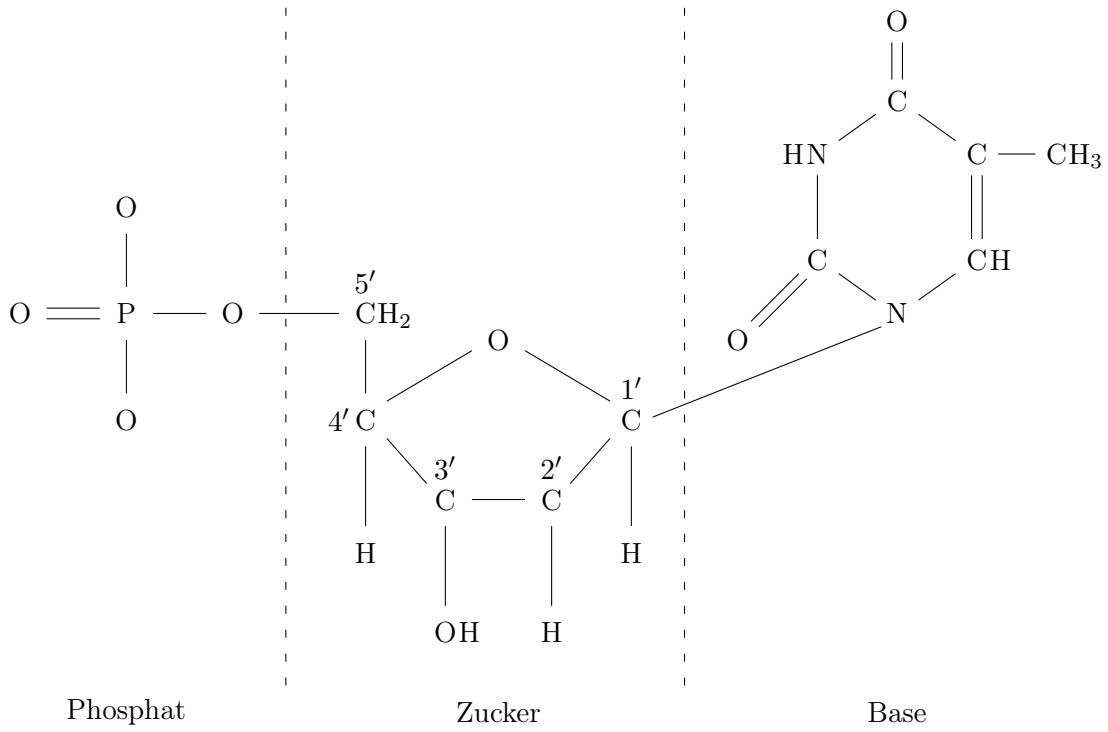

Abbildung 1: Chemische Struktur eines Nukleotids mit Base Thymin

zeigt ein schematisches Beispiel für beide Fälle.

3.1.2 Techniken zur Manipulation von DNA-Molekülen

Um mit DNA „rechnen“ zu können, muss man in der Lage sein, auf ihr „Rechenoperationen“ auszuführen. Dazu wenden wir uns nun einigen im *genetic engineering* angewandten Techniken zu, mittels derer man die Veränderung von DNA-Molekülen steuern kann.

Zuerst betrachten wir die Möglichkeit, DNA-Stränge durch spezielle Enzyme zu zerschneiden. Dazu verwendet man sogenannte *Endonukleasen*, die eine innere Phosphodiesterbindung aufspalten, wobei Ort und Art des Schnitts sehr unterschiedlich ausfallen können. Insbesondere seien an dieser Stelle die sogenannten *Restriktionsendonukleasen* (auch *Restriktionsenzyme*) erwähnt. Diese schneiden nur doppelsträngige DNA-Moleküle und zeichnen sich dadurch aus, dass die Spaltung an genau spezifizierten Stellen stattfindet. Hierbei stellt die Aufspaltung den „Urzustand“ der beteiligten Nukleotide wieder her, d. h. nach der Trennung liegt am 3′-Ende des einen Nukleotids die Hydroxygruppe vor und der Phosphatrest am 5′-Ende des anderen. Je nach eingesetztem Restriktions-

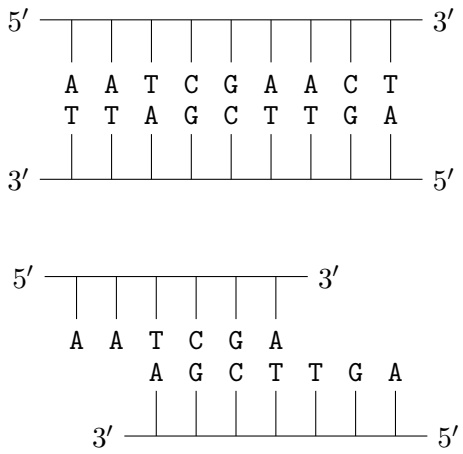

Abbildung 2: DNA-Moleküle mit *blunt* (oben) und *sticky ends* (unten)

enzym variiert nicht nur die Erkennungssequenz sondern auch die Form des Schnitts: manche Enzyme spalten geradewegs durch den Strang (produzieren also die bereits erwähnten „blunt ends“), wohingegen andere die Auftrennung leicht versetzt vornehmen. In letzterem Fall entstehen zwei Moleküle mit zueinander passenden „sticky ends“, wie in Abbildung 3 am Beispiel der Endonuklease *Eco*RI zu sehen. Die Erkennungssequenz lautet hier sowohl im oberen als auch unteren Strang 5′-`GAATTC`-3′ und der Schnitt findet nun wie angezeigt zwischen `G` und `A` statt, so dass jeweils ein überhängendes 5′-Ende bleibt.[2]

Wir stellen im Folgenden eine Technik vor, mittels derer solche Moleküle mit zueinander passenden, komplementären „sticky ends“ wieder zusammengefügt werden können. Einerseits reicht unter geeigneten Temperaturbedingungen die reine räumliche Nähe aus, um die komplementären Basen der Nukleotide eine Wasserstoffbrückenbindung eingehen zu lassen. Am 5′-Ende eines Strangs, das sich dann neben dem 3′-Ende des anderen Strangs befindet, bleibt jedoch genau wie im gegenüberliegenden Strang eine „Lücke“, da die Phosphodiesterbindung an diesen Stellen nicht besteht. Hier kommt nun die sogenannte *Ligation* zur Anwendung, bei welcher ein Enzym – entsprechend *Ligase* genannt – eben genau diese Bindungen katalysiert und die Enden somit fest miteinander verknüpft. Es sei angemerkt, dass die Ligation auch bei „blunt ends“ möglich ist, allerdings ist aufgrund der fehlenden Wasserstoffbrückenbindung die Wahrscheinlichkeit deutlich geringer, dass die beiden Enden dicht genug beieinander liegen.

[2] Der Fall, dass das Komplement einer einzelsträngigen Nukleotidfolge von hinten nach vorne gelesen identisch zur ursprünglichen Folge ist, wird als *palindromische Sequenz* bezeichnet und tritt bei den Erkennungssequenzen vieler Restriktionsenzyme auf.

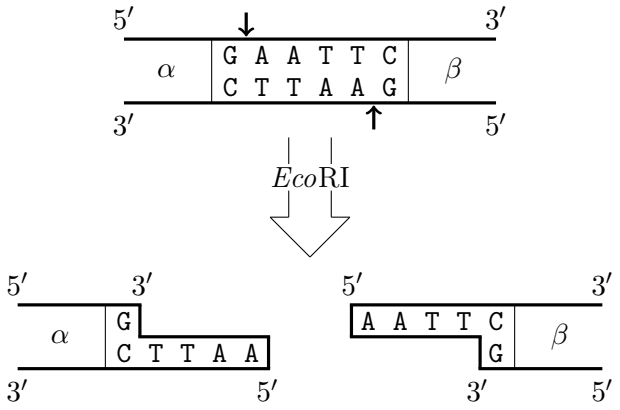

Abbildung 3: Spaltung eines Doppelstrangs durch *Eco*RI

Wie wir in den später vorgestellten Modellen sehen werden, liegt ihnen die Annahme zugrunde, dass der Vorrat an den jeweils benötigten DNA-Ketten nicht begrenzt ist, sondern stets „ausreichend viele“ zur Verfügung stehen. Daher soll nun eine Prozedur vorgestellt werden, mit der DNA vervielfältigt werden kann: die *Polymerase-Kettenreaktion*, kurz *PCR* (vom englischen *polymerase chain reaction*).

Zuerst sei hierfür noch einmal auf einen Unterschied zwischen den bisher betrachteten Bindungen hingewiesen: Während es sich bei der zwischen Phosphat- und Hydroxygruppe benachbarter Nukleotide auftretenden Phosphodiesterbindung um eine starke (kovalente) Bindung handelt, die auch sehr lange, einsträngige Kettenmoleküle zusammenhalten kann, ist die Wasserstoffbrückenbindung zwischen den Nukleinbasen verhältnismäßig schwach. Was der Verbindung der gegenüberliegenden Einzelstränge eines DNA-Moleküls ihre Stabilität verleiht, ist die Summe der zahlreichen zwischen ihnen gebildeten Wasserstoffbrücken.[3] Beim PCR-Vorgang erhitzt man nun die Lösung, welche die doppelsträngige DNA enthält, auf eine Temperatur nahe dem Siedepunkt, wodurch sich die Wasserstoffbrückenbindungen lösen und die beiden Einzelstränge voneinander trennen. Diesen Vorgang nennt man *Denaturierung*.

Anschließend wird beim sogenannten *Priming* die Temperatur auf ca. 55°C abgesenkt, was dazu führt, dass sich kurze einsträngige Sequenzen, die namensgebenden *Primer*, an die passenden, komplementären Stellen der DNA anlagern. Die Primer sind synthetisch hergestellte Ketten von Nukleotiden, die der Lösung hinzugefügt werden, um den Start der Bereiche festzulegen, die vervielfältigt werden sollen.

Zuletzt wird durch eine *Polymerase* ermöglicht, dass sich einzelne Nukleotide an den Primern beginnend anlagern, wobei durch die Komplementarität der Nukleotidbasen ge-

[3] Zwischen `C` und `G` bilden sich drei Wasserstoffbrückenbindungen aus im Gegensatz zu nur zweien zwischen `A` und `T`, weshalb die erstere Paarung etwas stabiler ist. Im Kontext der hier betrachteten Sachverhalte ist dieser Unterschied aber nicht weiter relevant.

währleistet ist, dass die neu entstandene Sequenz eine Kopie des Bereichs aus dem im ersten Schritt abgespalteten Strang ist. Diese Phase trägt den Namen *Extension*. Die Polymerase ermöglicht das Anlagern nur an der Hydroxygruppe eines freien 3′-Endes, weshalb das Priming notwendig ist, um den Kopiervorgang in Gang zu setzen.

Die Zusammensetzung der verwendeten Lösung sei hier zusammenfassend noch einmal nachgereicht: Sie enthält also neben den zu vervielfältigenden DNA-Strängen die entsprechenden Primer, DNA-Polymerase und die Nukleotide, die bei der Extension zum Erweitern der Stränge gebraucht werden. Die drei vorher angesprochenen Schritte werden so oft wiederholt, bis eine bestimmte Anzahl der gewünschten Moleküle erreicht ist, was durch das exponentielle Wachstum – die Anzahl der betreffenden DNA-Ketten verdoppelt sich mit jedem Durchlauf – auch für sehr große Mengen in kurzer Zeit erzielt werden kann.

3.1.3 DNA-Rekombination

Wir widmen uns an dieser Stelle noch einmal dem Fall, dass nach dem Zerschneiden mittels Restriktionsenzymen Doppelstränge mit zueinander passenden „sticky ends“ vorliegen. Wenn nun verschiedene Molekülketten durch Enzyme, die identische Überhangsequenzen erzeugen[4], zerschnitten werden, so lassen sich die entstandenen Fragmente auch „vertauscht“ wieder zusammenfügen.

Zur Verdeutlichung betrachten wir die folgenden DNA-Moleküle:

(1)
5′- G G G T C G A G G G -3′
3′- C C C A G C T C C C -5′

(2)
5′- T T T G C G C T T T -3′
3′- A A A C G C G A A A -5′

Dazu geben wir die Erkennungssequenz und Schnittposition der Enzyme *Taq*I und *Sci*NI an:

5′- T|C G A -3′	5′- G|C G C -3′
3′- A G C|T -5′	3′- C G C|G -5′
*Taq*I	*Sci*NI

Man beachte, dass beide den gleichen Überhang produzieren, nämlich 5′-CG-3′ mit überstehendem 5′-Ende. Die Erkennungssequenz von *Taq*I ist nun in (1) zu finden, die von *Sci*NI in (2). Wenn die Enzyme die DNA nun entsprechend zerschneiden, erhalten wir die folgenden Fragmente:

[4] D. h. sowohl Nukleotidfolge und Ausrichtung (entweder 5′- oder 3′-Ende überhängend) stimmen überein.

$$(1)\quad \begin{array}{l} 5'\text{-}\mathtt{GGGT}\quad\ \ \mathtt{CGAGGG}\text{-}3' \\ 3'\text{-}\mathtt{CCCAGC}\quad\ \ \mathtt{TCCC}\text{-}5' \end{array}$$

$$(2)\quad \begin{array}{l} 5'\text{-}\mathtt{TTTG}\quad\ \ \mathtt{CGCTTT}\text{-}3' \\ 3'\text{-}\mathtt{AAACGC}\quad\ \ \mathtt{GAAA}\text{-}5' \end{array}$$

Diese können nun bei Vorliegen einer Ligase entweder zu ihrer alten Form wiederhergestellt werden oder aber in neuer Konstellation zusammengefügt werden. Bei unserem Beispiel sind die vier Fragmente wegen der palindromischen Überhänge beliebig kombinierbar, im Allgemeinen ist nur die Möglichkeit der „kreuzweisen“ Rekombination gewährleistet, welche zu folgendem Ergebnis führt:

$$\begin{array}{l} 5'\text{-}\mathtt{GGGTCGCTTT}\text{-}3' \\ 3'\text{-}\mathtt{CCCAGCGAAA}\text{-}5' \end{array}$$

$$\begin{array}{l} 5'\text{-}\mathtt{TTTGCGAGGG}\text{-}3' \\ 3'\text{-}\mathtt{AAACGCTCCC}\text{-}5' \end{array}$$

Nach der Trennung wurde also das (wenn wir unsere Abbildung als Orientierung zulassen) „rechte“ Teilstück von (2) an das linke von (1) angefügt und andersherum. Es sei abschließend angemerkt, dass dieser Fall die zentrale Grundlage für den nächsten Abschnitt darstellt.

3.2 Formales Splicing

Da unsere Betrachtungen im Folgenden hauptsächlich doppelsträngige DNA-Moleküle beinhalten, wollen wir zunächst vereinbaren, diese nur noch durch Angabe *eines* Strangs zu repräsentieren, da dies den dazu komplementären ebenfalls eindeutig bestimmt. Zudem vereinbaren wir von nun an die $5'$-$3'$-Richtung als unsere (konstante) Leserichtung, so dass wir auch auf diese Angabe verzichten werden.

Die Erkennungssequenzen der zuletzt betrachteten Enzyme *Taq*I und *Sci*NI können wir demnach einfach mit `TCGA` und `GCGC` angeben. Wollen wir nun noch die beim Schneiden überlappenden Bereiche deutlich machen, so können wir dies durch die Angabe der Tripel $(\mathtt{T}, \mathtt{CG}, \mathtt{A})$ respektive $(\mathtt{G}, \mathtt{CG}, \mathtt{C})$ erreichen. Die Sequenz `CG` taucht nun im mittleren Glied beider Enzyme auf, wir wollen uns aber noch einmal bewusst machen, dass dies nur eine von zwei Bedingungen ist, die wir an die Kompatibilität von ‚sticky ends“ stellen: Neben der übereinstimmenden Nukleotidfolge muss dazu auch das gleiche Ende überhängen – also bei beiden entweder das mit freiem $5'$-Phosphatrest oder das mit freier $3'$-Hydroxygruppe. Sonst ist eine Rekombination der eben betrachteten Form nicht möglich.

Anstatt nun jedem der Tripel einen entsprechenden Vermerk hinzuzufügen, können wir stattdessen Paare von Tripeln der Form $((u_1, x, v_1), (u_2, x, v_2))$ angeben (mit $u_i, x, v_i \in$

$\{\mathtt{A}, \mathtt{T}, \mathtt{C}, \mathtt{G}\}^*, i \in \{1, 2\}$) und somit gewährleisten, dass innerhalb der durch die beschriebenen Enzyme entstandenen Fragmente eine Rekombination möglich ist. Seien also $w_1, w_2 \in \{\mathtt{A}, \mathtt{T}, \mathtt{C}, \mathtt{G}\}^*$ Beschreibungen von DNA-Strängen, wobei $w_1 = w_1' u_1 x v_1 w_1''$ und $w_2 = w_2' u_2 x v_2 w_2''$, dann erhalten wir als Resultat die Stränge $z_1 = w_1' u_1 x v_2 w_2''$ und $z_2 = w_2' u_2 x v_1 w_1''$. Diesen Vorgang bezeichnen wir als *Splicing* von w_1 und w_2.

Als nächstes wollen wir den Umstand, durch Blöcke von Nukleotiden auch eine größere Anzahl von Zeichen kodieren zu können, zum Anlass nehmen, die Beschränkung auf die vier Basensymbole aufzuheben und Zeichenreihen über beliebigen Alphabeten zuzulassen. Die Bedingungen, die eine solche Kodierung erfüllen muss, um auch einer Implementierung eines Problems im Labor zu genügen, sollen hier nicht weiter untersucht werden.[5] Man kann sich allerdings leicht vorstellen, dass z. B. bei einer naiven Kodierung eines 256-Zeichen-Vorrats durch feste Blöcke von je vier Nukleotiden jede Leserasterverschiebung zu unerwünschtem Verhalten führt.

Weiterhin wollen wir der in [PRS98] verwendeten Notation folgen (welche auch in [Fri08] aufgegriffen wird) und zur Darstellung der Splicing-Stellen geordnete Paare der Form (u, v) verwenden, da sich jedes der vorher erwähnten Tripel (u', x, v') auch in der Form (u', xv') oder $(u'x, v')$ darstellen lässt. Sei also V ein beliebig gewähltes Alphabet, $((u_1, u_2), (u_3, u_4))$ ein Paar von Erkennungsmustern ($u_i \in V^*, i \in [4]^+$) und ferner die Zeichenreihen $x = x_1 u_1 u_2 x_2$ und $y = y_1 u_3 u_4 y_2$ (mit $x_1, x_2, y_1, y_2 \in V^*$) gegeben. Dann ergibt das Splicing von x und y die Wörter $z = x_1 u_1 u_4 y_2$ und $w = y_1 u_3 u_2 x_2$.

In dieser Form wollen wir mittels Regeln die Bedingungen für das Splicing beschreiben:

Definition 3.1. Sei V ein Alphabet.

Eine *Splicing-Regel* über V ist ein Wort $r \in V^*\#V^*\$V^*\#V^*$, mit $\#, \$ \notin V$. Wenn $r = u_1\#u_2\$u_3\#u_4$ und $x = x_1 u_1 u_2 x_2, y = y_1 u_3 u_4 y_2$, mit $x_1, x_2, y_1, y_2 \in V^*$, so schreiben wir

$$(x_1 u_1 | u_2 x_2, y_1 u_3 | u_4 y_2) \vdash_r (x_1 u_1 | u_4 y_2, y_1 u_3 | u_2 x_2)$$

oder auch

$$(x, y) \vdash_r (z, w)$$

für $z = x_1 u_1 u_4 y_2, w = y_1 u_3 u_2 x_2$ und bezeichnen damit das *Splicing* von x und y. ■

Die vertikalen Balken in Definition 3.1 dienen lediglich der Übersicht und kennzeichnen, an welcher Stelle die Ausgangswörter geschnitten bzw. die Resultate wieder zusammengefügt werden. Auf die Motivation, Splicing-Regeln als Wörter der angegebenen Form darzustellen anstatt wie vorher als Paare von Paaren, soll später noch einmal eingegangen werden.

[5] In der verwendeten Literatur wird dieser Aspekt ebenfalls nicht behandelt, allerdings liegt der Schwerpunkt der Betrachtungen dort – wie in dieser Arbeit auch – nicht auf der praktischen Umsetzung der Modelle.

3.2.1 Splicing-Systeme

In diesem Abschnitt werden nun Systeme vorgestellt, welche die Splicing-Operation auf einer Menge von Zeichenreihen anwenden, und Betrachtungen bezüglich ihrer Berechnungsstärke vorgenommen. Dazu führen wir zunächst einige Begriffe ein:

Definition 3.2. Ein *H-Schema* ist ein Paar $\sigma = (V, R)$, wobei V ein beliebiges Alphabet ist und $R \subseteq V^*\#V^*\$V^*\#V^*$ eine Menge von Splicing-Regeln über V. ■

Definition 3.3. Für eine gegebenes H-Schema $\sigma = (V, R)$ und eine Sprache $L \subseteq V^*$ definieren wir das *One-Step-Splicing* von Wörtern aus L in Bezug auf R als

$$\sigma(L) = \{\ z \in V^* \mid (x, y) \vdash_r (z, w) \text{ oder } (x, y) \vdash_r (w, z),\ \text{für ein } x, y \in L \text{ und } r \in R\}.$$

■

Das H-Schema gibt also eine Menge von Regeln an (das Alphabet V wird aus formalen Gründen mit aufgeführt), mittels derer wir auf einer Menge von Zeichenreihen L die Splicing-Operation anwenden. Dabei geschieht eine solche Operation in einem Schritt immer parallel auf allen möglichen Kombinationen von Wörtern aus L, so wie Enzyme in einer Lösung mit ausreichendem Vorkommen jedes Moleküls ebenfalls eine große Anzahl von Strängen mit passender Sequenz zerschneiden und sich die Fragmente mit an Sicherheit grenzender Wahrscheinlichkeit in jeder möglichen Konstellation wieder zusammenfügen. Die der mathematischen Definition zugrundeliegende Vorschrift, dies synchron in Einzelschritten erfolgen zu lassen, ist dabei eine hilfsweise vorgenommene Annahme, die im folgenden Kontext aber keine ernste Einschränkung mehr darstellen wird. Zunächst wollen wir einen solchen Schritt an einem Beispiel veranschaulichen.

Beispiel 3.1. Sei $V = \{a, b, c\}, R = \{aa\#b\$b\#cc, c\#a\$bb\#c\}$ und $L = \{aabcca, abbcbb\}$. Wir erhalten

$$\sigma(L) = \{aacca, aabbcca, aabcccbb, abba\}$$

als Resultat des One-Step-Splicings.

Man beachte hierbei, dass x und y aus Definition 3.3 nicht verschieden sein müssen, im Beispiel ist entsprechend die erste Splicing-Regel auf zwei Exemplare des Wortes *aabcca* angewandt worden, woraus die Wörter *aacca* und *aabbcca* hervorgehen. Die anderen beiden erhalten wir als Ergebnis von

$$(aabcc|a, abb|cbb) \vdash_{c\#a\$bb\#c} (aabcc|cbb, abb|a).$$

Weitere Kombinationen sind nicht möglich. ■

Wenn in einem Reagenzglas Restriktionsenzyme und eine Ligase vorhanden sind, wirken diese mehr als nur einmal auf die vorhandenen Stränge inklusive der jeweils neu

gewonnenen Moleküle. Wir können uns diesen Prozess etwas vereinfacht dahingehend vorstellen, dass das Resultat eines One-Step-Splicings den ursprünglich vorhandenen Strängen hinzugefügt und auf diesem erweiterten Vorrat ein erneuter Schritt ausgeführt wird, so dass letztendlich alle möglichen Kombinationen auftreten. Formal sei dieser Vorgang wie folgt beschrieben:

Definition 3.4. Für ein gegebenes H-Schema $\sigma = (V, R)$ und eine Sprache $L \subseteq V^*$ definieren wir das *iterierte Splicing* von Wörtern aus L in Bezug auf R wie folgt:

$$\begin{aligned} \sigma^0 &= L, \\ \sigma^{i+1} &= \sigma^i(L) \cup \sigma(\sigma^i(L)),\ i \geq 0 \end{aligned}$$

und schließlich

$$\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L).$$

■

Nun können wir das für dieses Kapitel grundlegende Rechenmodell der Splicing-Systeme einführen. Im Prinzip unterscheidet sich dieses Modell nicht von dem in Definition 3.4 vorgestellten Mechanismus. Es arbeitet über einem Alphabet V, einer Menge von Splicing-Regeln R und einer Sprache $L \subseteq V^*$ und generiert bezüglich des somit spezifizierten H-Schemas $\sigma = (V, R)$ die Sprache $\sigma^*(L)$:

Definition 3.5. Ein *Splicing-* oder auch *H-System* ist ein Tupel $\gamma = (V, A, R)$, wobei V ein Alphabet, $A \subseteq V^*$ eine Sprache (wir nennen die Wörter aus A auch *Axiome*) und $R \subseteq V^*\#V^*\$V^*\#V^*$ eine Menge von Splicing-Regeln über V ist. Die *von γ erzeugte Sprache* $L(\gamma)$ ist definiert als

$$L(\gamma) = \sigma^*(A)$$

wobei $\sigma = (V, R)$ das γ zugrunde liegende H-Schema ist. ■

[PRS98] beschäftigt sich intensiv mit der Berechnungsstärke von H-Systemen. Dabei wurde konkret untersucht, inwieweit die Klasse der von einem H-System $\gamma = (V, A, R)$ generierten Sprache $L(\gamma)$ von den Klassen abhängt, aus denen die Sprachen A und R stammen. In diesem Kontext führen wir die folgende Bezeichnung ein:

Definition 3.6. Für zwei Klassen von Sprachen $\mathcal{K}_A, \mathcal{K}_R$ definieren wir die Klasse

$$\mathcal{H}(\mathcal{K}_A, \mathcal{K}_R) = \{\sigma^*(A) \mid A \in \mathcal{K}_A,\ \sigma = (V, R) \text{ für } R \in \mathcal{K}_R \text{ und beliebige Alphabete } V\}.$$

■

Nach Definition 3.5 gilt zudem

$$\mathcal{H}(\mathcal{K}_A, \mathcal{K}_R) = \{L(\gamma) \mid \gamma = (V, A, R) \text{ für } A \in \mathcal{K}_A, R \in \mathcal{K}_R\}.$$

An dieser Stelle soll noch einmal Bezug auf die im Zuge von Definition 3.1 geänderte Repräsentation einer Splicing-Regel aus R von einem Paar von Paaren $((u_1, u_2), (u_3, u_4))$ in eine Zeichenreihe der Form $u_1\#u_2\$u_3\#u_4$ genommen werden. Durch diese Definition ist die Menge R nun nämlich selbst eine Sprache und kann z. B. einer der Chomsky-Hierarchie entnommenen Klasse zugeordnet werden, was zusammen mit Definition 3.6 klare Aussagen über den Einfluss der Komplexität der Regeln auf die von entsprechenden Splicing-Systemen erzeugte Sprachklasse ermöglicht.

Da man unter realen Umständen nur auf eine endliche Menge von DNA-Strängen und einen begrenzten Vorrat an Restriktionsenzymen zurückgreifen kann, ist für das DNA-Computing hierbei vor allem der Fall interessant, bei dem A und R endlich sind. In [PRS98] wird nun gezeigt, dass in diesem Fall gilt:

$$FIN \subset \mathcal{H}(FIN, FIN) \subset REG$$

Weiterhin wird gezeigt, dass die Klasse der rekursiv aufzählbaren Sprachen von H-Systemen nur dann erzeugt wird, wenn wir rekursiv aufzählbare Menge von Axiomen verwenden (vgl. [PRS98], S. 238, Tabelle 7.2.). Das Resultat wird etwas verbessert, wenn wir nur noch Wörter über einem Alphabet von Terminalsymbolen als Ergebnis zulassen, wie wir es auch von Grammatiken formaler Sprachen kennen. Da wir in Abschnitt 5 nach dem gleichen Prinzip nur bestimmte Wörter als Ausgabe akzeptieren, soll die erweiterte Definition hier ebenfalls vorgestellt werden:

Definition 3.7. Ein *erweitertes H-System* ist ein Tupel $\gamma = (V, T, A, R)$, wobei V ein Alphabet, $T \subseteq V$ eine Menge von *Terminalsymbolen*, $A \subseteq V^*$ eine Menge von Axiomen und $R \subseteq V^*\#V^*\$V^*\#V^*$ eine Menge von Splicing-Regeln über V ist. Die von γ erzeugte Sprache $L(\gamma)$ ist definiert als

$$L(\gamma) = \sigma^*(A) \cap T^*$$

mit $\sigma = (V, R)$. ■

Dazu geben wir an dieser Stelle zwei weitere Resultate aus [PRS98] an:

$$(1) \qquad \mathcal{EH}(FIN, FIN) = REG.$$

$$(2) \qquad \mathcal{EH}(FIN, REG) = RE.$$

Dabei ist $\mathcal{EH}(\mathcal{K}_A, \mathcal{K}_R)$ analog zu Definition 3.6 die Klasse der von erweiterten H-Systemen erzeugten Sprachen in Abhängigkeit der Klassen der Axiom- und Splicing-Regel-Menge.

Nach (2) reicht bereits die Verfügbarkeit von regulären Mengen von Splicingregeln aus, um alle rekursiv aufzählbaren Sprachen generieren zu können. Dies ist allerdings

angesichts nur endlich vieler Restriktionsenzyme ein für die Praxis unrealistisches Szenario. Insofern sehen wir uns der unbefriedigenden Situation ausgesetzt, dass ein nach dem Prinzip erweiterter H-Systeme arbeitender DNA-Rechner nicht über die Berechnungsstärke eines endlichen Automaten hinauskommt. In diesem Kontext sei auch auf die ebenfalls in [PRS98] vorgestellte Tatsache hingewiesen, dass es keinen „universellen“ endlichen Automaten gibt in dem Sinne eines endlichen Automaten F_u, der als Eingabe eine Kodierung eines beliebigen endlichen Automaten F' und dessen Eingabe a' erhält und diese genau dann akzeptiert, wenn a' in der von F' akzeptierten Sprache liegt. Dies bedeutet, dass wir auf der Basis erweiterter H-Systeme auch keinen universellen, d. h. programmierbaren DNA-Rechner konstruieren können.

Dies begründet die Motivation, Splicing-Systeme mit membranbasierten Rechenmodellen zu kombinieren, so dass wir zu den später betrachteten Splicing-P-Systemen gelangen. Wir werden sehen, dass diese Systeme Möglichkeiten der Steuerung über die als Resultat der Splicing-Operation erhaltenen Zeichenreihen bieten, die es uns erlaubt, auch mit endlichen Ressourcen universelle Berechnungsstärke zu erlangen. Dazu sollen Membransysteme zunächst einmal in ihrer ursprünglichen Form vorgestellt werden.

3.3 Bibliografische Anmerkungen

Die hier verwendete Definition von Splicing-Regeln wird in [PRS98] eingeführt, wo jedoch zusätzlich zwischen 1- und 2-Splicing unterschieden wird, je nachdem, ob man beim Splicing zweier Wörter nur eine der bei der Rekombination entstehenden Zeichenreihen als Resultat betrachtet oder beide. Während wir hier 2-Splicing-Operationen der Form $(x, y) \vdash_r (z, w)$ verwenden, beschränken sich die Untersuchungen dort zunächst auf 1-Splicing der Form $(x, y) \vdash_r z$. Dies beinhaltet auch alle Definitionen und Resultate zu H-Systemen. Da aber jedes 2-Splicing von x und y in Bezug auf eine Splicingregel $r = u_1\#u_2\$u_3\#u_4$ auch durch zwei 1-Splicing-Operationen beschrieben werden kann, stellt dies keine Einschränkung des Modells dar. So liefert, wenn im obigen Kontext $(x, y) \vdash_r (z, w)$ gilt, das 1-Splicing der Form $(x, y) \vdash_r z$ und das der Form $(y, x) \vdash_{r'} w$ (wobei $r' = u_3\#u_4\$u_1\#u_2$) letztendlich das gleiche Ergebnis. Dementsprechend wird auch in der erwähnten Quelle später gezeigt, dass alle Resultate nahezu unverändert auf das 2-Splicing übertragbar sind.

Die in diesem Kapitel vorgenommenen Betrachtungen zu Splicing-Systemen folgen somit denen in [PRS98], legen jedoch gleich das 2-Splicing zugrunde. Da in einem realen System zu erwarten ist, dass beide Ergebnisse des Splicings mit gleicher Wahrscheinlichkeit vorliegen, wird dies hier als die realistischere Beschreibung betrachtet und somit auf eine Behandlung von 1-Splicing außerhalb dieser Anmerkungen verzichtet.

4 Membrane Computing

Nachdem wir mit dem Splicing Reaktionen betrachtet haben, die *in vitro* (lat. für „im Glas“, d. h. in einer künstlich geschaffenen Umgebung wie z. B. einem Reagenzglas) umgesetzt werden, wollen wir uns nun einer Klasse von Modellen widmen, die den in lebenden Zellen (*in vivo*) stattfindenden Vorgängen nachempfunden wurden. Eine Zelle weist trotz ihrer mikroskopisch kleinen Größe eine komplexe Struktur auf, die den Ablauf zahlreicher Prozesse ermöglicht. Sie verfügt dazu über spezifische „Funktionseinheiten“, wie beispielsweise Nukleus, Mitochondrien oder endoplasmisches Retikulum in eukaryotischen Zellen, die von Membranen umgeben sind oder aus Membranen bestehen. Auch die Zelle als Ganzes ist von einer äußeren Membran (der *Zell-* oder auch *Plasmamembran*) umschlossen.

Die Membranen sind dabei in mehrfacher Hinsicht von Bedeutung: Einerseits grenzen sie die einzelnen Bestandteile räumlich voneinander ab und stellen somit einen Bereich zur Verfügung, in welchem Reaktionen isoliert ablaufen können. Andererseits bieten sie aber auch Kanäle zur Kommunikation und zum Stofftransport zwischen den einzelnen Regionen. Sie spielen eine zentrale Rolle bei der Zellfunktion und deren Steuerung und sind daher Gegenstand von Untersuchungen geworden, die ein auf diesem Prinzip aufbauendes „Rechnermodell“ anstreben, die sogenannten *Membransysteme*. In Analogie zu den H-Systemen hat sich hierbei aber auch der Begriff *P-System* etabliert (nach G. Păun, der diese eingeführt hat).

Wegen der großen strukturellen Nähe von Membransystemen zu biologischen Zellen gibt es auch Bestrebungen, P-Systeme zu einer möglichst exakten Modellierung einer Zelle und der in ihr ablaufenden Prozesse zu verwenden (vgl. z. B. [BGK+06], [CPP06]). Păun betont jedoch in [PR02] selbst, dass dies im Allgemeinen nicht das Ziel ist, sondern die Natur vielmehr als Inspirationsquelle dient, verschiedene Eigenschaften abstrahiert zu übertragen und deren Auswirkungen auf das Modell in Bezug auf Berechenbarkeit zu untersuchen. So liegt der Schwerpunkt auch in dieser Arbeit auf dem mathematischen Modell; die biologischen Hintergründe werden im nächsten Abschnitt nur ausschnitthaft angedeutet (vgl. hierzu auch [Pău02], ein Überblick findet sich ebenso in [PRS10]).

4.1 Die Plasmamembran

Da P-Systeme eine sehr abstrakte Repräsentation darstellen, wollen wir uns auf die Darstellung einiger Eigenschaften biologischer Membranen beschränken, die als Motivation für die Mechanismen in P-Systemen verstanden werden können. Uns genügt dabei die Vorstellung, dass Membranen einerseits eine Abgrenzung von Reaktionsräumen darstellen und andererseits der Kommunikation zwischen diesen Räumen dienen.

Nach dem sogenannten *Flüssig-Mosaik-Modell*, welches 1972 von S. Singer und G. Nicolson vorgeschlagen wurde, ist eine Membran eine im Wesentlichen aus Phospholipiden bestehende Doppelschicht, in welche unter anderem Proteine eingebunden sind. Diese

können sich – ebenso wie die Lipide – innerhalb ihrer Membranschicht frei bewegen, woher auch die Bezeichnung „flüssig“ im Namen des Modells rührt. Die Proteine sind an dieser Stelle deshalb für uns interessant, da sie eine wichtige Rolle beim Transmembran-Transport von Stoffen spielen.

Dieser vollzieht sich z. B. mittels sogenannter *Ionenkanäle*. Das sind Transmembranproteine (d. h. beide Ebenenen der Doppellipidschicht durchquerend), welche geladenen Teilchen das Durchqueren der Membran entlang des elektrochemischen Gradienten (also entlang des Konzentrationsgefälles) ermöglichen. Der Transport kann dabei je nach Protein selektiv für bestimmte Ionen erfolgen, wobei der Kanal dementsprechend meist nach diesen benannt ist. So lässt sich z. B. ein Kaliumkanal für Kaliumionen um ein Vielfaches leichter durchqueren als für andere Ionen. Außerdem können sich die Kanäle unter bestimmten Umwelteinflüssen schließen bzw. wieder öffnen.

Ionen können aber auch entgegen dem Konzentrationsgefälle durch die Membran transportiert werden, nämlich wiederum mit Beteiligung bestimmter Proteine, den sogenannten *Ionenpumpen*. Die Bewegung in diese Richtung benötigt Energie, die durch die enzymgesteuerte Spaltung von ATP (Adenosintriphosphat, dem Energieträger in Zellen) bereitgestellt wird. Man bezeichnet dies auch als *aktiven Transport*.

Weiterhin existiert die Möglichkeit des Transports unter Beteiligung von *Carrier-Proteinen*. Dieser Vorgang kann an einen zweiten Transportprozess gekoppelt sein, bei dem ein weiteres Teilchen simultan entweder in der gleichen Richtung durch die Membran befördert wird (*Symport*) oder das Durchqueren in entgegengesetzten Richtungen verläuft (*Antiport*). Den Transport eines einzelnen Teilchens, der katalytisch durch ein Carrier-Protein ermöglicht wird, bezeichnet man als *Uniport*.

Wir wollen festhalten, dass die Permeabilität bei Membranen hochgradig selektiv ausgeprägt ist, d. h. dass also nur ganz bestimmte Teilchen unter bestimmten Bedingungen (z. B. dem Vorhandensein eines Katakysators) passieren können. Wir werden diese Eigenschaft bei der Definition von P-Systemen wiederfinden.

4.2 P-Systeme

Die Membranstruktur eines P-Systems ist eine hierarchische Anordnung von Membranen. Wir können uns in diesem Modell eine Membran als dreidimensionale Blase vorstellen, die wiederum weitere Membranen enthalten kann. Dabei gibt es genau eine äußere Membran, die *Oberflächenmembran*, welche direkt oder indirekt alle weiteren Membranen enthält. Die Membranstruktur lässt sich somit graphentheoretisch durch einen Baum repräsentieren, bei welchem die Oberflächenmembran den Wurzelknoten darstellt und alle direkt enthaltenen Membranen jeweils einen Nachfolgerknoten bilden. Membranen, die keine weiteren Membranen enthalten und somit den Blättern des repräsentierenden Baumes entsprechen, werden *elementar* genannt.

Um die einzelnen Membranen referenzieren zu können, werden wir sie mit natürlichen Zahlen bezeichnen, was zumeist eine Nummerierung von 1 bis m ($m \in \mathbb{N}^+$) bedeutet, wo-

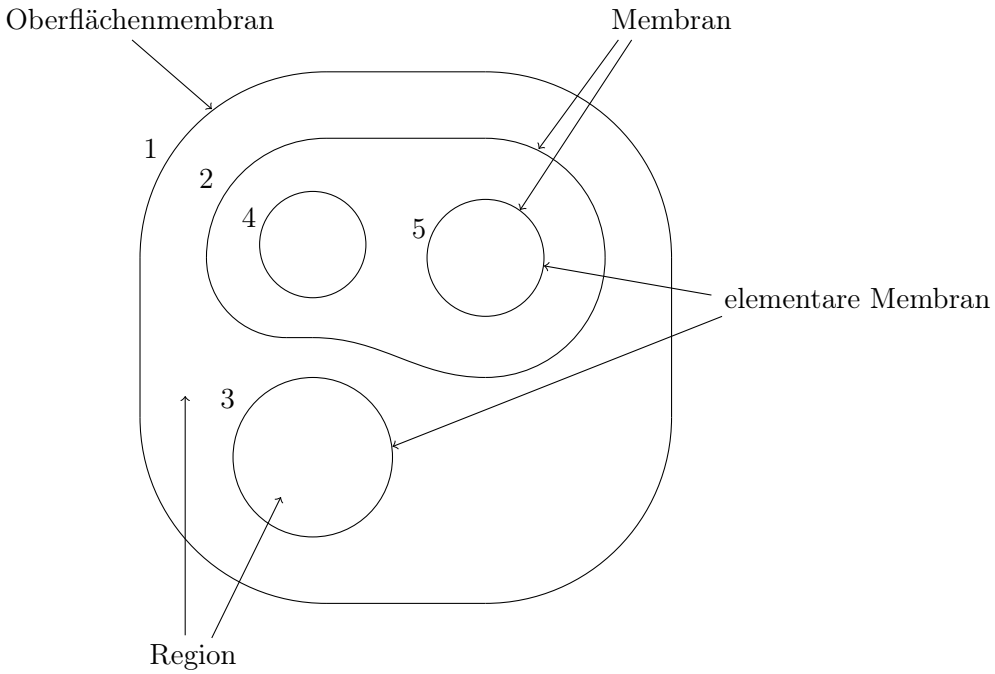

Abbildung 4: Beispiel einer Membranstruktur

bei 1 die Oberflächenmembran kennzeichnet. Jede Membran bestimmt zudem genau eine *Region*. Darunter verstehen wir den Raum, der direkt durch eine Membran eingeschlossen ist, d. h. zwischen der Membran und eventuell direkt in dieser liegenden weiteren Membranen. Umgekehrt ist somit auch jeder Region eindeutig ihre Membran zuzuordnen, weshalb wir die Bezeichner der Membranen gleichzeitig auch für ihre jeweiligen Regionen verwenden. In Abbildung 4 findet sich die Darstellung einer Beispielstruktur. Für den allgemeinen Fall geben wir nun eine formale Beschreibung an. Sie basiert auf der in [Fri08] verwendeten Repräsentation.

Definition 4.1. Eine *Membranstruktur* ist ein Baum $\mu = (Q, E)$, wobei $Q \subset \mathbb{N}^+$ eine endliche Menge von Knoten ist, und $E \subseteq Q \times Q$ die Menge der Kanten.

Wir nennen $i \in Q$ *Membran*. Eine Membran j heißt *elementar*, wenn es kein $i \in Q$ gibt, so dass $(j, i) \in E$. Außerdem bezeichnet $m = |Q|$ den *Grad* von μ. ■

Eine Berechnung eines P-Systems geschieht in den einzelnen Regionen auf *Objekten*, welche wir durch Zeichen über einem *Objektalphabet* repräsentieren, und zwar durch die Anwendung sogenannter *Evolutionsregeln*. Diese Regeln sind für jede Region individuell angegeben. Die Objekte, die als Ergebnis der Anwendung entstehen, können dann innerhalb ihrer Membran verweilen oder in eine angrenzende Region übergehen. Letzteres bedeutet entweder einen Übergang aus der Membran heraus in die sie umgebende

Region bzw. die das gesamte P-System beinhaltende *Umgebung*, falls sie aus der Oberflächenmembran abgegeben werden, oder den Transport in eine in der Region direkt enthaltenen Membranen hinein. Wir werden dies kennzeichnen, indem wir die durch eine Evolutionsregel erzeugten Objekte mit einem der Zielindikatoren *here*, *out* oder in_j versehen. Dabei ist j die Bezeichnung einer inneren Membran, wir können also bei mehreren solcher Membranen direkt eine bestimmte als Ziel adressieren.

Definition 4.2. Eine *Evolutionsregel* ist ein Paar (u, v), auch notiert als $u \to v$, wobei $u \in V^*$, V ein Alphabet und $v \in V_{tar}^*$ ist mit $V_{tar} = \{\ a_\tau \mid a \in V, \tau \in tar\}, tar = \{here, out, in_j \mid j \in \mathbb{N}^+\}$.

Der *Radius* der Regel $u \to v$ ist $|u|$. ■

In der rechten Seite v einer solchen Regel erlauben wir es uns, den Zielindikator *here* nicht mit zu notieren.

Bei der Anwendung einer Evolutionsregel $u \to v$ in einer Region werden alle in u notierten Objekte aus der Region entfernt und die auf der rechten Seite v der Regel vorkommenden Objekte neu erschaffen und gemäß ihres Zielindikators auf die Ausgangs- oder eine der angrenzenden Regionen verteilt. Die Objekte, die sich zu einem Zeitpunkt in einer Region befinden, charakterisieren wir mit Multimengen, die jeder Region zugeordnet sind. Da wir Objekte als Zeichen aus einem Alphabet V darstellen, so können wir die Multimengen von Objekten innerhalb einer Region dementsprechend als Wörter über V beschreiben. Auch die Menge der Evolutionsregeln, welche die Art und Bedingungen der jeweils möglichen Reaktionen beschreiben, wird individuell für jede Membran angegeben.

In einem Berechnungsschritt werden in jeder Region Evolutionsregeln *maximal parallel* auf die mit den Regionen assoziierten Multimengen angewandt, d. h. solange es Objekte gibt, auf die noch keine Regel angewandt wird und diese der linken Seite u einer Evolutionsregel entsprechen, muss diese (oder eine andere passende Regel) ausgeführt werden. Wir werden diesen Vorgang später noch einmal detaillierter darlegen. Es findet hierbei eine Parallelität auf zwei Ebenen statt, nämlich einerseits in Form der gleichzeitigen, maximalen Anwendung von Regeln auf die Objekte in einer Region und andererseits durch die gleichzeitige Bearbeitung aller Regionen des P-Systems.

Die Berechnung eines P-Systems endet, wenn nach einem Schritt in keiner Region weitere Evolutionsregeln anwendbar sind. Als Gesamtresultat verstehen wir dann die Objekte, die das System während der Berechnung verlassen haben, d. h. aus der Oberflächenmembran an die Umgebung abgegeben wurden. Bevor wir dies an einem Beispiel veranschaulichen, wollen wir eine formale Beschreibung angeben:

Definition 4.3. Ein *P-System* ist ein Tupel

$$\Pi = (V, \mu, w_{q_1}, \ldots, w_{q_m}, R_{q_1}, \ldots, R_{q_m}).$$

Dabei gilt:

- V ist ein Alphabet von *Objekten*
- $\mu = (Q, E)$ ist eine Membranstruktur vom Grad m mit $Q = \{q_1, \ldots, q_m\}$ als Menge der Membranen
- $w_{q_i} \in V^*, 1 \leq i \leq m$, sind Repräsentationen von Multimengen über V, die mit der Region $q_i \in Q$ assoziiert sind
- $R_{q_i}, 1 \leq i \leq m$, sind endliche Mengen von Evolutionsregeln, die mit der Region $q_i \in Q$ assoziiert sind

Wir nennen m den *Grad von* Π, er entspricht dem Grad von μ. ■

Für eine Membranstruktur $\mu = (Q, E)$ vom Grad m wollen wir ohne Beschränkung der Allgemeinheit davon ausgehen, dass $Q = \{1, \ldots, m\}$. Die allgemeinere Form aus Definition 4.1 liegt darin begründet, dass es erweiterte P-Systeme gibt, bei denen auch die Möglichkeit des Auflösens einer Membran besteht. Daraus entstehende Membranstrukturen können somit ebenfalls als definitionsgemäß betrachtet werden.

Definition 4.4. Sei $\Pi = (V, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m)$ ein P-System.
Eine *Konfiguration* von Π ist ein m-Tupel $(v_1, \ldots, v_m)$ mit $v_i \in V^*, i \in [m]^+$. Wir bezeichnen $(w_1, \ldots, w_m)$ als *initiale Konfiguration* von Π. ■

Für ein P-System $\Pi = (V, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m)$ und zwei Konfigurationen $C_1 = (w'_1, \ldots, w'_m), C_2 = (w''_1, \ldots, w''_m)$ von Π nennen wir $C_1 \Rightarrow C_2$ eine *Transition* von C_1 zu C_2, wenn C_2 durch maximal parallele Anwendung von Evolutionsregeln aus R_i auf w'_i entsteht (für alle $i \in \{1, \ldots, m\}$). Dies soll hier noch einmal genauer beschrieben werden.

Betrachten wir eine Regel $u \to v \in R_i$. Die Anwendung dieser Evolutionsregel in der Region i bedeutet nun, dass wir die Zeichen, die in der linken Seite u vorkommen, aus der Multimenge w_i entfernen. Die Regel ist also nur anwendbar, wenn alle Symbole aus u auch in w_i vorhanden sind.

Danach werden für alle a_τ, die in der rechten Seite v der Evolutionsregel vorkommen, die entsprechenden Objekte a den Multimengen hinzugefügt, die durch die Membranstruktur μ und $\tau \in \{here, out, in_j \mid j \in [m]^+\}$ angegeben sind. D. h. also:

- Falls $\tau = here$, wird a zur ursprünglichen Region w_i hinzugefügt.
- Falls $\tau = out$, wird a der benachbarten äußeren Region hinzugefügt. Dies entspricht dem Elterknoten j von i in $\mu = (Q, E)$, der bei einem Baum für alle Knoten außer der Wurzel eindeutig bestimmbar ist, d. h. für $(j, i) \in E$ fügen wir a der mit dieser Region assoziierten Multimenge w_j hinzu. Falls i die Oberflächenmembran ist, wird a an die Umgebung abgegeben. Wir können uns dazu die Membranstruktur auch um einen Knoten 0 erweitert vorstellen, welcher die Umgebung bezeichnet, wobei zu E die Kante $(0, i)$ für die Oberflächenmembran i hinzugefügt wird. Dann können

wir mit w_0 die Multimenge der Objekte angeben, die in die Umgebung abgegeben wurden. Evolutionsregeln werden auf w_0 selbstverständlich nicht angewandt, ein R_0 existiert nicht.

- Falls $\tau = in_j$, so muss, damit die Regel angewendet werden kann, eine Kante $(i, j) \in E$ existieren, d. h. j ist direkt in i enthalten. Wir fügen dann a zu w_j hinzu. In elementaren Membranen sind Evolutionsregeln dieser Form demnach niemals anwendbar.

Das Hinzufügen von Objekten gemäß der rechten Regelseiten geschieht dabei erst am Ende eines Berechnungsschrittes, also nachdem für jede angewandte Regel die linken Seiten entfernt wurden. Die neuen Objekte ermöglichen eine eventuelle weitere Regelanwendung somit erst bei der nächsten Transition. Die Auswahl der Regeln in einem Schritt geschieht nichtdeterministisch, d. h. wenn mehrere Regeln angewendet werden können (jedoch nicht gleichzeitig, da ihre linken Seiten um die gleichen Objekte konkurrieren), so wird eine beliebige dieser Regeln ausgewählt. Solange noch Objekte in einer Multimenge vorhanden sind, also noch keiner linken Seite einer Regel zugewiesen wurden, und diese die Anwendung von weiteren Regeln ermöglichen, muss dies wegen der Forderung maximaler Parallelität auch erfolgen.

Eine *Berechnung* eines P-Systems Π ist nun eine Folge von Konfigurationen $C_1 \Rightarrow C_2 \Rightarrow \ldots$, wobei die Berechnung genau dann *erfolgreich* ist, wenn sie hält, d. h. es existiert ein $n \in \mathbb{N}$ und eine Berechnung $C_1 \Rightarrow \ldots \Rightarrow C_n$, so dass auf die Objekte in C_n keine weitere Regel anwendbar ist.

Wir wollen nun noch einmal genauer darauf eingehen, was wir unter der Ausgabe eines P-Systems verstehen. Hierbei werden stets nur die Objekte betrachtet, die im Zuge einer erfolgreichen, also haltenden, Berechnung an die Umgebung abgegeben werden. Zum einen besteht die Möglichkeit, nur die Gesamtanzahl dieser Objekte als Resultat zu verstehen (analog zu [Pău02]). Sei also w_0 die Multimenge der Objekte, die in der letzten Konfiguration der Berechnung in der Umgebung enthalten sind, dann ergibt die Berechnung von Π das Resultat $|w_0|$. Die Menge der Zahlen, die von Π berechnet werden kann, bezeichnen wir dann mit $N(\Pi)$. Wir verwenden auch die Formulierung, dass die Menge $N(\Pi)$ *von* Π *erzeugt* wird.

Eine andere Möglichkeit besteht in einer etwas aussagekräftigeren Repräsentation von w_0, bei der die Häufigkeit der verschiedenen Symbole $a \in V$ in der Ausgabemenge angegeben wird. Dann ist (vgl. [PR02]) das Resultat der Berechnung der Vektor $\Psi_V(w_0)$. Die Menge aller von Π berechenbaren Ergebnisvektoren bezeichnen wir mit $Ps(\Pi)$ und werden analog zu vorher von der von Π erzeugten Menge $Ps(\Pi)$ sprechen.

Falls ein P-System Π Evolutionsregeln mit einem Radius größer als eins enthält, bezeichnen wir es als ein System *mit Kooperation* oder auch *kooperatives* System, anderenfalls dementsprechend als *nicht-kooperativ*. Kooperative Systeme verfügen somit über die Fähigkeit, ähnlich zu Grammatiken formaler Sprachen kontextsensitive Bedingungen für

die Anwendung von Regeln festzulegen, was – wie wir sehen werden – auch für P-Systeme unter dem Gesichtspunkt der Berechenbarkeit zu einer erhöhten Mächtigkeit des Systems führt.

An dieser Stelle soll die Funktionsweise eines P-Systems anhand eines Beispiels veranschaulicht werden:

Beispiel 4.1. Sei $\Pi = (V, \mu, w_1, w_2, R_1, R_2)$ ein P-System (mit Grad 2), wobei

- $V = \{c, s\}$
- $\mu = (\{1, 2\}, \{(1, 2)\})$
- $w_1 = s$
- $w_2 = \varepsilon$
- $R_1 = \{s \to ss_{in_2}s_{in_2},\ cc \to c_{out}c_{out}s_{out},\ s \to c_{out}\}$
- $R_2 = \{s \to c_{out}, sss \to s_{out}\}$

In der initialen Konfiguration findet sich als einziges Objekt ein Vorkomen von s innerhalb der Oberflächenmembran 1, wir können also nur die erste Regel anwenden. Diese schickt zwei Exemplare von s in Region 2 und stellt ein s in 1 wieder zur Verfügung. Alle s, die aus 1 in Region 2 geschickt werden, kommen in der nächsten Transition als c wieder zurück in die Oberflächenmembran. Von dort werden sie gemäß der zweiten Regel entfernt und an die Umgebung abgegeben, zusammen mit einem neu erzeugten Vorkommen von s. Man beachte, dass in R_2 die erste Regel wegen der maximal parallelen Anwendung stets auf beide Vorkommen von s angewendet wird und die Voraussetzungen für die zweite Regel somit niemals erfüllt sind.

Die Berechnung endet, nachdem wir in Region 1 die Regel $s \to c_{out}$ anwenden, wonach wir keine Vorkommen von s mehr im System haben und somit spätestens nach einer weiteren Transition (in der die eventuell aus 2 geschickten c noch an die Umgebung abgegeben werden) keine Regeln mehr anwendbar sind.

Dann ist in w_0 wegen der eben betrachteten letzten Regel aus 1 mindestens ein Vorkommen von c enthalten. Es ist zudem festzustellen, dass jede Anwendung der ersten Regel in 1 genau eine Anwendung der zweiten Regel $cc \to c_{out}c_{out}s_{out}$ in der übernächsten Transition nach sich zieht. Wenn nun also n die Anzahl der Anwendungen der ersten Regel aus R_1 bezeichnet, führt dies zu weiteren $2n$ Vorkommen von c und n Exemplaren von s in der Umgebung. Für den Fall $n = 3$ sieht die Berechnung wie folgt aus (zur Konfiguration von Π ist auch die Ausgabemenge w_0 mit angegeben):

$$(s, \varepsilon), w_0 = \varepsilon \Rightarrow (s, ss), \varepsilon \Rightarrow (ccs, ss), \varepsilon \Rightarrow (ccs, ss), c^2s \Rightarrow (cc, \varepsilon), c^4cs^2 \Rightarrow (\varepsilon, \varepsilon), c^6cs^3.$$

Wir erhalten als Ausgabe $|w_0| = 10$ bzw. $\Psi_V(w_0) = (7, 3)$. Die von Π erzeugten Mengen sind $N(\Pi) = \{3n + 1 \mid n \in \mathbb{N}\}$ und $Ps(\Pi) = \{(2n + 1, n) \mid n \in \mathbb{N}\}$. ■

4.2.1 Mächtigkeit von P-Systemen

Wir wollen in diesem Kapitel Resultate über die Berechnungsstärke von P-Systemen angeben. Dazu werden wir nicht nur P-Systeme in der bisher eingeführten Form betrachten, sondern auch kurz einige modifizierte Varianten vorstellen. Dies soll lediglich mit der Motivation erfolgen, einen Überblick über die Leistungsfähigkeit von P-Systemen zu geben, weshalb wir auf die Angabe von Beweisen verzichten und ein Verweis auf die jeweilige Quelle genügen soll.

Beginnen wollen wir mit P-Systemen in ihrer grundlegenden Form, wie sie in Definition 4.3 angegeben sind. Dazu bezeichnen wir mit $\mathcal{NP}_m(ncoo)$ die Klasse der Mengen $N(\Pi)$ für nicht-kooperative P-Systeme Π mit maximalem Grad m. Falls der Grad nicht begrenzt ist, notieren wir dies mit $*$. Analog schreiben wir $\mathcal{NP}_m(coo)$ für die entsprechende Klasse für P-Systeme mit Kooperation.

Die nächsten beiden Resultate stammen aus [Pău02]:

$$\mathcal{NP}_*(ncoo) = \mathcal{NP}_1(ncoo) = NCF.$$

Nicht-kooperative Systeme erlangen somit durch eine vergrößerte Anzahl von verfügbaren Membranen keine erhöhte Stärke der Berechenbarkeit und können nicht mehr als die Längenmengen kontextfreier Sprachen generieren.

Lassen wir Kooperation zu, verbessert sich das Ergebnis zu

$$\mathcal{NP}_*(coo) = \mathcal{NP}_m(coo) = NRE, m \geq 1.$$

Auch hierbei stellt die Verwendung nur einer Membran gegenüber Systemen höheren Grades keine signifikante Einschränkung dar. Die Kontextsensitivität der Evolutionsregeln bietet bereits einen ausreichend mächtigen Mechanismus, der zur Berechnung aller rekursiv aufzählbaren Mengen von natürlichen Zahlen auf das Vorhandensein mehrerer Membranen nicht angewiesen ist. Insofern ist auch die Untersuchung von eingeschränkteren Varianten interessant.

Hierunter fallen z. B. die sogenannten *katalytischen* Systeme, in denen eine Teilmenge $C \subseteq V$ von *Katalysatoren* gegeben ist, welche bei der Anwendung mancher Evolutionsregeln vorhanden sein müssen und nach der Ausführung wieder unverändert vorliegen (in Analogie zu Katalysatoren bei chemischen Prozessen). Die einzigen Regeln mit Radius größer als eins sind dabei von der Form $cu \to cv$, wobei $c \in C, u \in V \setminus C$ und $v \in (V \setminus C)^*_{tar}$ mit $(V \setminus C)_{tar}$ analog Definition 4.2, Vorkommen von Katalysatoren $c \in C$ in u und v sind also nicht erlaubt. In allen Regeln anderer Form, d. h. mit Radius eins, dürfen ebenfalls keine Elemente aus C auftauchen.

Wir bezeichnen nun mit $\mathcal{NP}_m(cat)$ die Klasse der von katalytischen P-Systemen m-ten Grades erzeugten Mengen. Erwartungsgemäß liegt diese Klasse „zwischen“ den Klassen von nicht-kooperativen Systemen und denen kooperativer Systeme. So gilt nach Definition der jeweils zugelassenen Evolutionsregeln offensichtlich (vgl. auch [Pău02]):

$$\mathcal{NP}_*(ncoo) \subseteq \mathcal{NP}_*(cat) \subseteq \mathcal{NP}_*(coo).$$

Welche der hierbei auftretenden Teilmengenbeziehungen echt sind, bleibt in der Quelle (und somit auch hier) ungelöst, dort wird zumindest eine Ungleichheit der ersten beiden Klassen vermutet.

Eine wiederum etwas erweiterte Variante stellt die Verwendung von *bistabilen* Katalysatoren dar, d. h. dass jeder Katalysator zwei Zustände (c und $\overline{c}$) besitzt. Bei der Anwendung katalytischer Evolutionsregeln muss sich c im korrekten Zustand befinden, wechselt diesen im Zuge dessen aber stets. Die katalytischen Regeln sind demnach von der Form $cu \to \overline{c}v$ respektive $\overline{c}u \to cv$. Diese Erweiterung reicht aus, um auch mit nur einer Membran rekursiv aufzählbare Ergebnismengen erzeugen zu können. Die Klasse der von P-Systemen Π mit bistabilen Katalysatoren erzeugten Mengen $Ps(\Pi)$ bezeichnen wir mit $\mathcal{NP}_s\mathcal{P}_m(2cat)$. Der hierbei erfolgte Wechsel bei der Betrachtung der Ausgabe zu einer Menge von Vektoren natürlicher Zahlen ist durch die entsprechende Behandlung in [PR02] motiviert, wo sich das erwähnte Resultat finden lässt, nämlich:

$$PsRE = \mathcal{NP}_s\mathcal{P}_1(2cat).$$

Zwei weitere Modifikationen sollen ebenfalls erwähnt sein: Dies ist zum einen das Einführen je einer Prioritätsrelation ρ_i auf den Mengen der Evolutionsregeln R_i, was bedeutet, dass eine Regel nur dann in einer Region ausgeführt werden darf, wenn keine Regel höherer Priorität anwendbar ist.[6] Zum anderen kann man Evolutionsregeln erlauben, die mit ihnen assoziierte Membran aufzulösen, was durch ein spezielles Symbol $\delta \notin V$ gekennzeichnet wird. Kommt eine solche Regel zur Anwendung, wird die Membran aus der Membranstruktur zusammen mit ihrer Regelmenge entfernt. Die in der Region vorhandenen Objekte gehen dabei in die äußere Region über, wo nachfolgend die zu dieser Membran gehörenden Evolutionsregeln auf sie angewendet werden können. Das Auflösen der Oberflächenmembran ist nicht erlaubt.

Beide Erweiterungen stellen eine Möglichkeit dar, zusätzliche Bedingungen für die Anwendung von Evolutionsregeln festzulegen, weswegen sie jeweils auch eine erhöhte Klasse berechenbarer Mengen mit sich bringen können. Auf die genaue Angabe weiterer Ergebnisse zur Mächtigkeit solcher Systeme verzichten wir, da diese Modifikationen bei unserer folgenden Betrachtung der Splicing-P-Systeme nicht zum Einsatz kommen.

Anhand der ausgewählten Resultate soll nur veranschaulicht sein, welchen Einfluss eine Erweiterung der Kontextsensitivität in den Regeln auf die Mächtigkeit der Klasse der erzeugbaren Ergebnismengen hat, da im nächsten Abschnitt entsprechende Betrachtungen unter einem ähnlichen Aspekt vorgenommen werden.

[6] ρ_i ist dabei eine strenge Halbordnung, d. h. irreflexiv und transitiv, allerdings nicht total, so dass bei der Definition zweier Regeln der Menge R_i kein „Zwang“ zur Priorisierung einer der beiden besteht.

5 Splicing-P-Systeme

5.1 Einführung

Splicing-P-Systeme stellen eine Kombination von H- und P-Systemen dar, und zwar insofern, als dass sie Splicing-Regeln innerhalb einer Membranstruktur verwenden. Die Evolutionsregeln arbeiten hierbei nicht mehr auf Objekten, sondern sie sind Splicing-Regeln, die auf einer jeder Region zugeordneten Axiomenmenge zur Anwendung kommen. Dazu wird für beide durch eine Splicing-Operation erzeugten Wörter mit einem Zielindikator angegeben, in welchen Regionen diese schließlich vorliegen. Neben der Menge der Axiome verfügt jede Membran auch über eine eigene Menge von Evolutionsregeln.

Analog zu Splicing-Systemen gehen wir auch hier davon aus, dass die Axiome in unendlich großer Anzahl vorliegen. Im Gegensatz zu P-Systemen, bei denen die auf der linken Seite einer Evolutionsregel vorkommenden Objekte aus der Region entfernt werden, bleiben die Axiome hier in jeder Region erhalten. Als Ausgabe betrachten wir dann alle Wörter über einem Alphabet von Terminalsymbolen, wie wir es von erweiterten H-Systemen kennen, die an die Umgebung abgegeben werden.

Der Eindeutigkeit halber wollen wir hier in Abgrenzung zu Definition 4.2 die Bezeichnung *Splicing-Evolutionsregel* einführen, werden uns aber erlauben, auch weiterhin einfach von *Evolutionsregeln* zu sprechen, wenn durch den Kontext klar ist, dass wir Splicing-P-Systeme betrachten.

Definition 5.1. Ein *Splicing-P-System* ist ein Tupel

$$\Pi = (V, T, \mu, L_{q_1}, \ldots, L_{q_m}, R_{q_1}, \ldots, R_{q_m}),$$

wobei gilt:

- V ist ein Alphabet mit $\#, \$ \notin V$.
- $T \subseteq V$ ist ein Alphabet von Terminalsymbolen.
- $\mu = (Q, E)$ mit $Q = \{q_1, \ldots, q_m\}$ ist eine Membranstruktur, wobei jedes $q \in Q$ eine Region in Π bezeichnet.
- $L_i \subseteq V^*$ ist eine endliche Sprache von Axiomen, die mit der Region i assoziiert ist, für alle $i \in Q$.
- R_i ist eine mit Region i assoziierte Menge von *Splicing-Evolutionsregeln* der Form $(r; tar_1, tar_2)$, wobei $r \in V^*\#V^*\$V^*\#V^*$ eine Splicing-Regel ist und $tar_1, tar_2 \in \{here, in, out\}$, für alle $i \in Q$.

$m \in \mathbb{N}^+$ bezeichnet den *Grad* von Π. Der *Durchmesser* von Π ist definiert als

$$dia(\Pi) = (p_1, p_2, p_3, p_4),$$

wobei $p_i = \max(\{|u_i| \mid (u_1\#u_2\$u_3\#u_4; tar_1, tar_2) \in \bigcup_{q\in Q} R_q\}), 1 \leq i \leq 4$. Falls eines der p_i unendlich ist, so ersetzen wir es mit $*$. ■

Definition 5.2. Sei $\Pi = (V, T, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m)$ ein Splicing-P-System.
Eine *Konfiguration* von Π ist ein m-Tupel $(M_1, \ldots, M_m)$ mit $M_i \subseteq V^*, i \in [m]^+$. Wir bezeichnen $(L_1, \ldots, L_m)$ als *initiale Konfiguration* von Π. ■

Eine Evolutionsregel $r = (u_1\#u_2\$u_3\#u_4; tar_1, tar_2) \in R_i$, $i \in Q$ ermöglicht hierbei mit $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_3 \in M_i$, das Splicing $(x, y) \vdash_r (z_1, z_2)$. Die Wörter z_1 und z_2 werden dabei entsprechend Zielindikator tar_1 bzw. tar_2 und $\mu = (Q, E)$ wie folgt behandelt (für $k \in \{1, 2\}$):

- Wenn $tar_k = here$, dann wird z_k den Axiomen M_i der ursprünglichen Region i hinzugefügt.
- Wenn $tar_k = in$, dann geht z_k genau dann zu M_j über, wenn es ein $j \in Q$ gibt mit $(i, j) \in E$. Falls i atomar ist, kann r nicht angewendet werden.[7]
- Wenn $tar_k = out$, dann fügen wir z_k der mit der äußeren Region j assoziierten Menge M_j hinzu, d. h. für $(j, i) \in E$. Falls i die Oberflächenmembran ist, so wird z_k an die Umgebung abgegeben. Für diesen Fall wollen wir uns die Umgebung wieder als Region 0 vorstellen, mit $(0, i) \in E$. Dementsprechend werden wir mit M_0 die Sprache der Wörter bezeichnen, die das System bis zum betrachteten Zeitpunkt verlassen haben und bei Relevanz diese Menge auch in einer Konfiguration von Π mit aufführen.

Für zwei Konfigurationen $C = (M_0, M_1, \ldots, M_m)$, $C' = (M'_0, M'_1, \ldots, M'_m)$ eines Splicing-P-Systems Π bezeichnen wir mit $C \Rightarrow C'$ eine *Transition* von C nach C' und meinen damit die maximal parallele Anwendung aller Evolutionsregeln aus R_i auf M_i für alle $i \in Q$, so dass wir danach $(M'_0, M'_1, \ldots, M'_m)$ vorliegen haben. Dementsprechend erhalten wir $M'_0 = M_0 \cup M_{out}$, wobei M_{out} die Menge der im Zuge dieser Transition an die Umgebung abgegebenen Wörter ist.

Unter einer *Berechnung* von Π verstehen wir auch hier wieder eine Sequenz von Transitionen $C_1 \Rightarrow C_2 \Rightarrow \ldots$, allerdings ist im Gegensatz zu P-Systemen eine ausschließliche Betrachtung *haltender* Berechnungen, in dem Sinne, dass keine weitere Evolutionsregel mehr angewendet werden kann, nicht sinnvoll. Jede initiale Konfiguration lässt entweder keine einzige Regelanwendung zu oder es sind unendlich lange Berechnungen möglich, da keine Axiome entfernt werden und somit nach einer Transition alle erfolgten Regelanwendungen erneut vorgenommen werden können. Als Ausgabe gelten alle Wörter über dem Terminalalphabet, die von dem System an die Umgebung abgegeben werden.

[7] Man beachte, dass hier auf die Möglichkeit einer direkten Adressierung einer bestimmten inneren Membran verzichtet wurde und z_k somit durch die maximal parallele Anwendung letztendlich allen direkt benachbarten inneren Regionen hinzugefügt wird.

Wir definieren für ein Splicing-P-System $\Pi = (V, T, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m)$ mit initialer Konfiguration $C_0 = (\emptyset, L_1, \ldots, L_m)$ die *von* Π *erzeugte Sprache*

$$L(\Pi) = \{w \in M_0 \cap T^* \mid \exists n \in \mathbb{N}^+ \colon \exists C_n = (M_0, \ldots, M_m) \colon C_0 \Rightarrow \ldots \Rightarrow C_n\}.$$

Dies soll an dieser Stelle an einem kurzen Beispiel veranschaulicht werden.

Beispiel 5.1. Sei $\Pi_1 = (V, T, \mu, L_1, L_2, L_3, R_1, R_2, R_3)$ ein Splicing-P-System, wobei gilt:

- $V = \{A, B, c\}$
- $T = \{c\}$
- $\mu = (\{1, 2, 3\}, \{(1, 2), (2, 3)\})$
- $L_1 = \{B\}$
- $L_2 = \{AcA, B\}$
- $L_3 = \emptyset$
- $R_1 = \{(A\#\varepsilon\$\varepsilon\#B; out, out)\}$
- $R_2 = \{(A\#c\$c\#A; in, here), (c\#A\$B\#\varepsilon; out, in)\}$
- $R_3 = \emptyset$

In Region 2 führt die Anwendung von $r_{2,1} = (A\#c\$c\#A; in, here)$ zu $(A|cA, Ac|A) \vdash_{r_{2,1}} (AA, AccA)$, wobei AA in die innere Region 3 wandert und $AccA$ in 2 bleibt. Es zeigt sich, dass somit im Laufe einer Berechnung in Region 2 Wörter der Form Ac^nA entstehen, wegen $(A|c^kA, Ac^l|A) \vdash_{r_{2,1}} (AA, Ac^{k+l}A)$ für $n, k, l \geq 1$. Eine Anwendung von $r_{2,2} = (c\#A\$B\#\varepsilon; out, in)\}$ führt zu $(Ac^n|A, B|\varepsilon) \vdash r_{2,2}(Ac^n, BA)$, wobei Ac^n in Region 1 übergeht und BA nach innen.

Region 3 erfüllt dabei keine wirkliche Funktion, weshalb sie auch über keine Regeln verfügt, sondern dient uns lediglich als Container, in welchen die bei obigen Regelanwendungen entstandenen „Nebenprodukte“ AA bzw. BA gehen.

In Region 1 kommen die Wörter der Form Ac^n an, was dann folgendes Splicing ermöglicht: $(A|c^n, \varepsilon|B) \vdash_{(A\#\varepsilon\$\varepsilon\#B; out, out)} (AB, c^n)$. Beide Splicing-Resultate werden dabei an die Umgebung abgegeben, allerdings zählt nur c^n als Ausgabe des Systems, da $AB \notin T^*$.

Eine konkrete Berechnung sieht wie folgt aus:

$$(\emptyset, \{B\}, \{AcA, B\}, \emptyset)$$

$$\Rightarrow (\emptyset, \{Ac, B\}, \{AcA, AccA, B\}, \{AA, BA\})$$

$$\Rightarrow (\{c, AB\}, \{Ac, Acc, B\}, \{AcA, AccA, AcccA, AccccA, B\}, \{AA, BA\})$$

$$\Rightarrow (\{c, cc, AB\}, \{Ac, Acc, Accc, Acccc, B\}, \{Ac^nA, B \mid n \in [2^3]^+\}, \{AA, BA\})$$

$$\Rightarrow (\{c^n, AB \mid n \in [2^2]^+\}, \{Ac^n, B \mid n \in [2^3]^+\}, \{Ac^nA, B \mid n \in [2^4]^+\}, \{AA, BA\}).$$

Das System hat den Durchmesser $dia(\Pi_1) = (1, 1, 1, 1)$ und erzeugt die Sprache $L(\Pi_1) = \{c^n \mid n \in \mathbb{N}^+\}$.

Das Beispiel veranschaulicht zudem, wie die parallele Regelanwendung ein exponentielles Wachstum der Ausgabemenge im Verhältnis zur „Laufzeit“ des Systems ermöglicht, so liegt jedes Wort $w \in \{c^n \mid n \in [2^k]^+\}$ nach $k + 2$ Transitionen vor (für $k \in \mathbb{N}$). ■

Die Klasse der Sprachen $L(\Pi)$, die von Splicing-P-Systemen Π mit maximalem Grad m, einer Membranstrukturtiefe[8] von höchstens d und einem Durchmesser von höchstens (p_1, p_2, p_3, p_4), erzeugt werden können, bezeichnen wir mit $\mathcal{SPL}_m(d, p_1, p_2, p_3, p_4)$.

5.2 Rotate-and-Simulate

Bevor wir zu den Betrachtungen über die Mächtigkeit dieser Klassen übergehen, wollen wir eine Technik erläutern, die oftmals in den dazu vorgenommenen Beweisen verwendet wird.[9] Dabei wird zu einer beliebigen Grammatik G eines der Chomsky-Hierarchie entnommenen Typs ein Splicing-P-System Π konstruiert, das die Produktionen von G simuliert, d. h. $L(G) = L(\Pi)$. Da $L(\Pi) \in \mathcal{SPL}_m(d, p_1, p_2, p_3, p_4)$, für entsprechende m, d, p_1, p_2, p_3, p_4, ist somit relativ einfach gezeigt, dass diese Klasse von durch Splicing-P-Systeme erzeugten Sprachen mindestens so mächtig ist wie die entsprechende Klasse, aus der G stammt.

Sei also $G = (N, T, S, P)$ eine Grammatik mit $\beta \to \gamma \in P$, $\beta, \gamma \in (N \cup T)^*$. Ferner seien $w = w_1\beta w_2$, $w' = w_1'\beta w_2'$ Satzformen von G mit $w_1, w_2, w_1', w_2' \in (N \cup T)^*$, $w_1 \neq w_1', w_2 \neq w_2'$, so dass gilt $w \Rightarrow_{\beta\to\gamma} z$ und $w' \Rightarrow_{\beta\to\gamma} z'$, d. h. $z = w_1\gamma w_2$ und $z' = w_1'\gamma w_2'$.

Will man nun mit einem entsprechenden Splicing-P-System Π die Anwendung der Regel $\beta \to \gamma$ simulieren, könnte man dies versuchen, indem man zuerst w und w' in die Teilwörter $w_1\beta, w_2, w_1'\beta$ und w_2' spaltet, und anschließend mittels einer entsprechenden Splicing-Regel β durch γ ersetzt, es lägen also $w_1\gamma, w_2, w_1'\gamma$ und w_2' vor. Weiterhin könnte man Π ermöglichen, diese Teilwörter wieder zusammenzufügen, so dass z und z' entstehen. Allerdings gilt es dabei zu verhindern, dass w_2 und w_2' an das γ des jeweils anderen ersten Teilwortes angehängt werden, da die Wörter $w_1\gamma w_2'$ und $w_1'\gamma w_2$ keine Satzform von G darstellen könnten. Da dies mit endlich vielen Splicing-Regeln im Allgemeinen nicht sichergestellt werden kann, da die Kontexte w_1, w_2 bei unendlichen Sprachen eine unbegrenzte Länge aufweisen können, verwenden wir eine andere Vorgehensweise, bei der die Wörter rotiert werden.

[8] Damit ist die Tiefe eines Baumes gemäß Definition in Kapitel 2 gemeint, wobei die Membranstruktur hierbei ohne die bisweilen vorgenommene Erweiterung um einen Knoten 0 für die Umgebung betrachtet wird.

[9] vgl. hierzu auch [Fri08], S. 240 ff.

Definition 5.3. Sei V ein beliebiges Alphabet und $u \in V^*$. Die Menge $cycle(u)$ der *Rotationen* von u ist die kleinste Menge, die folgende Eigenschaften erfüllt:

1. $u \in cycle(u)$
2. Wenn $u'\alpha \in cycle(u)$ ($u' \in V^*$, $\alpha \in V$), dann $\alpha u' \in cycle(u)$.

■

Lässt man nun in Π das Wort $w_1\beta w_2$ k-mal schrittweise „vorwärts“ rotieren für $k = |w_2|$, so erhalten wir $w_2w_1\beta$. Darauf kann nun eine Splicing-Regel die Anwendung der Produktionsregel zu $w_2w_1\gamma$ simulieren. Dieses formen wir anschließend wieder durch l Rotationsschritte, $l = |w_1\gamma|$, zu $w_1\gamma w_2$ um, welches eine Satzform von G ist. Die Produktion der „irrtümlichen“ Satzform $w_1\gamma w_2'$ ist dabei nicht mehr möglich, und wir benötigen zur Repräsentation einer Produktionsregel $\beta \to \gamma$ nur eine Splicing-Regel, die β am Ende eines Wortes entfernt und durch γ ersetzt. Diese Verfahrensweise trägt dementsprechend den Namen *Rotate-and-Simulate*.

Π muss nun bei den Rotationen von $w_2w_1\gamma$ erkennen können, wann es sich im korrekten Ausgangszustand $w_1\gamma w_2$ befindet, ohne dass wir dabei Bedingungen an die Kontexte w_1, w_2 stellen, da dieses bei unendlich vielen möglichen Kontexten die Notwendigkeit unendlich vieler Regeln nach sich ziehen würde. Dazu werden wir den Anfang des ursprünglichen Wortes mit einem neuen, in den Kontexten somit nicht enthaltenen Symbol $B \notin V$ kennzeichnen. Wir beginnen demnach mit $Bw_1\beta w_2$ und rotieren dieses, bis die linke Seite der simulierten Produktionsregel am rechten Ende steht, es liegt also $w_2Bw_1\beta$ vor. Dann ist die Anwendung der ersetzenden Splicing-Regel möglich, deren Ergebnis $w_2Bw_1\gamma$ wir anschließend wieder rotieren, bis das Symbol B am linken Ende des Wortes steht und dieses letztendlich in der gewünschten Form $Bw_1\gamma w_2$ vorliegt.

Um nun das Vorhandensein bestimmter Symbolfolgen an den Worträndern zu erkennen, wie z. B. das Vorliegen von B am Anfang eines Wortes oder der linken Seite β einer Produktionsregel an dessem rechten Ende, bedienen wir uns noch weiterer Hilfszeichen. Diese kennzeichnen den linken bzw. rechten Rand eines Wortes und werden nicht mitrotiert.

Beispiel 5.2. Wir geben hier die Definition eines Splicing-P-Systems, welches alle Rotationen eines Wortes β der Länge k über einem Alphabet von n Symbolen erzeugt, d. h. $\beta = \beta_1 \dots \beta_k$, $\beta_i \in \{a_1, \dots, a_n\}$, $i \in [n]^+$ und $k, n \geq 2$.

Sei $\Pi_2 = (V, T, \mu, L_1, L_2, R_1, R_2)$ ein Splicing-P-System, wobei:

- $V = \{X, Y, W, Z\} \cup \{X_i, Y_i, X'_{i-1}, Y'_{i-1} \mid i \in [n]^+\} \cup T$
- $T = \{a_1, \dots, a_n\}$
- $\mu = (\{1, 2\}, \{(1, 2)\})$

- $L_1 = \{X\beta Y\} \cup \{ZY_i, X_iZ, X_ia_iZ \mid i \in [n]^+\} \cup \{ZY, XZ, W\}$
- $L_2 = \{ZY_i', X_i'Z \mid i \in [n-1]\}$
- $R_1 = \{r_{1,i} \mid r_{1,i} = (\varepsilon\#a_iY\$Z\#Y_i; here, out), i \in [n]^+\} \cup$
  $\{r_{2,i,j} \mid r_{2,i,j} = (X_ia_i\#Z\$X\#a_j; in, out), i,j \in [n]^+\} \cup$
  $\{r_{3,i} \mid r_{3,i} = (a_i\#Y_0'\$Z\#Y; here, out), i \in [n]^+\} \cup$
  $\{r_{4,i} \mid r_{4,i} = (X\#Z\$X_0'\#a_i; here, out), i \in [n]^+\} \cup$
  $\{r_{5,i} \mid r_{5,i} = (a_i\#Y\$W\#\varepsilon; here, out), i \in [n]^+\} \cup$
  $\{r_{6,i} \mid r_{6,i} = (\varepsilon\#W\$X\#a_i; out, out), i \in [n]^+\} \cup$
  $\{r_{7,i} \mid r_{7,i} = (\varepsilon\#Y_i'\$Z\#Y_i; here, out), i \in [n-1]^+\} \cup$
  $\{r_{8,i,j} \mid r_{8,i,j} = (X_i\#Z\$X_i'\#a_j; in, out), i \in [n-1]^+, j \in [n]^+\}$
- $R_2 = \{r_{9,i,j} \mid r_{9,i,j} = (a_j\#Y_i\$Z\#Y_{i-1}'; out, here), i,j \in [n]^+\} \cup$
  $\{r_{10,i,j} \mid r_{10,i,j} = (X_{i-1}'\#Z\$X_i\#a_j; here, here), i,j \in [n]^+\}$

Die Hilfssymbole X, Y kennzeichnen den linken und rechten Rand eines Wortes, dementsprechend liegt in Region 1 anfangs $X\beta Y$ vor. Alle zyklischen Permutationen β' von β werden später ebenfalls in der Form $X\beta' Y$ vorliegen. Sei $\beta' = \beta_1' \dots \beta_k'$ mit $\beta_1' = a_i$ und $\beta_k' = a_j$ für je ein $i, j \in [n]^+$. Dann kann das folgende Splicing ausgeführt werden:

$$(X\beta_1' \dots \beta_k'|Y, W|\varepsilon) \vdash_{r_{5,j}} (\underbrace{X\beta_1' \dots \beta_k'}_{here}, \underbrace{WY}_{out}) \text{ und anschließend}$$

$$(X|\beta_1' \dots \beta_k', \varepsilon|W) \vdash_{r_{6,i}} (\underbrace{XW}_{out}, \underbrace{\beta_1' \dots \beta_k'}_{out}).$$

Dabei zählt nur β' zur Ausgabe, da alle anderen an die Umgebung abgegebenen Wörter Symbole enthalten, die nicht aus T stammen. So kann man die Regel $r_{6,i}$ auch zuerst anwenden, es gilt dann ebenfalls offensichtlich $\beta_1' \dots \beta_k' Y \notin L(\Pi_2)$.

Für ein Wort $X\beta_1' \dots \beta_k' Y$ wird nun die nächste zyklische Permutation $X\beta_k'\beta_1' \dots \beta_{k-1}' Y$ auf folgende Art erzeugt ($\beta_k' = a_j$ für ein $j \in [n]^+$):

(i) $\beta_k' = a_j$ wird am rechten Ende abgeschnitten, wobei das System die Information darüber, welches Symbol aus T entfernt wurde, in dem a_j zugeordneten Endsymbol Y_j behält: $(X\beta_1' \dots \beta_{k-1}'|\beta_k' Y, Z|Y_j) \vdash_{r_{1,j}} (\underbrace{X\beta_1' \dots \beta_{k-1}' Y_j}_{here}, \underbrace{Z\beta_k' Y}_{out})$
$Z\beta_k' Y$ zählt dabei wieder nicht zu $L(\Pi_2)$.

(ii) Im so entstandenen $X\beta_1' \dots \beta_{k-1}' Y_j$ wird X durch X_ia_i ersetzt, für alle $i \in [n]^+$:
$(X_ia_i|Z, X|\beta_1' \dots \beta_{k-1}' Y_j) \vdash_{r_{2,i,m}} (\underbrace{X_ia_i\beta_1' \dots \beta_{k-1}' Y_j}_{in}, \underbrace{XZ}_{out}), \beta_1' = a_m$

Die Wörter $X_i a_i \beta'_1 \dots \beta'_{k-1} Y_j$ wandern in Region 2. Wir müssen nun noch die zyklischen Permutationen herausfiltern, d. h. genau die Wörter, in denen $i = j$ gilt und somit am Wortanfang genau das Symbol hinzugefügt wurde, welches am rechten Ende abgeschnitten worden war.

(iii) In Region 2 werden nun nacheinander die Regeln $r_{10,i,i}$ und $r_{9,j,o}$ angewendet (mit $\beta'_{k-1} = a_o$):
$(X'_{i-1}|Z, X_i|a_i\beta'_1 \dots \beta'_{k-1}Y_j) \vdash_{r_{10,i,i}} (\underbrace{X'_{i-1}a_i\beta'_1 \dots \beta'_{k-1}Y_j}_{here}, \underbrace{X_iZ}_{here})$ und
$(X'_{i-1}a_i\beta'_1 \dots \beta'_{k-1}|Y_j, Z|Y'_{j-1}) \vdash_{r_{9,j,o}} (\underbrace{(X'_{i-1}a_i\beta'_1 \dots \beta_{k-1}Y'_{j-1}}_{out}, \underbrace{ZY_j}_{here})$
Die Start- und Endsymbole X_i, Y_j werden hierbei also durch $'$-markierte Versionen ersetzt und ihr Index um eins dekrementiert. Das Resultat geht zurück in die äußere Region. Die nebenher entstehenden Wörter X_iZ, ZY_j verbleiben in 2, ohne dort zum Gegenstand einer Splicing-Regel werden zu können.

(iv) In Region 1 werden nun Wörter der Form $X'_p a_s \dots a_t Y'_q$ mit $p, q \in [n-1]^+, s, t \in [n]^+$ wiederum durch Anwendung von $r_{7,q}$ und $r_{8,p,s}$ zu $X_p a_s \dots a_t Y_q$ umgewandelt und in Region 2 geschickt. Von dort kehren sie nach einem Schritt (iii) entsprechenden Vorgang als $X'_{p-1} a_s \dots a_t Y'_{q-1}$ zurück. Dies wiederholt sich solange, bis eines der Enden den Index 0 aufweist. Falls ein solches Wort ursprünglich in Schritt (ii) mit $i = j$ zustandegekommen ist, d. h. dort in der Form $X_j\beta'_k\beta'_1 \dots \beta'_{k-1}Y_j$ für $a_j = \beta'_k$ erzeugt wurde, liegt nun also $X'_0\beta'_k\beta'_1 \dots \beta'_{k-1}Y'_0$ vor.

(v) Die Regeln $r_{3,i}$ und $r_{4,o}$ (mit $a_i = \beta'_k, a_o = \beta'_{k-1}$) ersetzen nun X'_0 durch X bzw. Y'_0 durch Y. In Region 1 ist somit das Wort $X\beta'_k\beta'_1 \dots \beta'_{k-1}Y$ vorhanden und der Rotationsschritt ist abgeschlossen.

Für das in der initialen Konfiguration vorliegende $X\beta Y$ erzeugt Π_2 in Region 1 somit letztendlich alle Wörter $X\beta' Y, \beta' \in \mathit{cycle}(\beta)$, wobei β' (wie anfangs beschrieben) an die Umgebung abgegeben wird. ■

In dieser Art lässt sich zu einer Grammatik $G = (N, T, S, P)$ nun ein Splicing-P-System Π_G mit identischem Ausgabealphabet T konstruieren, welches alle Satzformen w von G in der Form $XBwY$ erzeugt, $X, Y, B \notin N \cup T$, ausgehend von einem initialen Vorkommen von $XBSY$. Dabei existiert für jede Produktionsregel $\beta \to \gamma \in P$ in Π_G eine simulierende Evolutionsregel r der Form $(\#\beta Y\$Z\#\gamma Y, here, out)$ zusammen mit entsprechenden Vorkommen von $Z\gamma Y, Z \notin (N \cup T)$. Diese kommt immer genau dann zum Einsatz, wenn ein Wort mit Suffix βY vorliegt:

$$(Xw_2Bw_1|\beta Y, Z|\gamma Y) \vdash_r (\underbrace{Xw_2Bw_1\gamma Y}_{here}, \underbrace{ZY}_{out}), w_1, w_2 \in (N \cup T)^*.$$

Die dazu eventuell notwendigen Rotationen liefert ein inneres Teilsystem nach der im obigen Beispiel vorgestellten Funktionsweise, so dass für jedes $X\beta Y$, $\beta \in (N \cup T \cup \{B\})^*$ auch alle $X\beta' Y$, $\beta' \in cycle(\beta)$ in der Region vorhanden sind. Alle gültigen Satzformen lassen sich dann wiederum am Präfix XB erkennen. Das System muss bei diesen Wörtern der Form $XBwY$ nun lediglich die Worträndern abtrennen und w an die Umgebung abgeben. Es gilt dann: $w \in L(G)$ gdw. $w \in T^*$ gdw. $w \in L(\Pi_G)$ und somit $L(G) = L(\Pi_G)$.

Beispiel 5.2 ist eine Erweiterung eines in [Fri08] gegebenen Splicing-P-Systems mit $T = \{a, b, c\}$ auf Ausgabealphabete beliebiger Größe. Ein ähnlicher Mechanismus wird in Abschnitt 6 in einem anderen Zusammenhang noch einmal Verwendung finden.

5.3 Betrachtungen zur Berechenbarkeit

In [Fri08] finden sich einige Resultate bezüglich der Beschreibungsstärke von Splicing-P-Systemen. Dabei werden auch einige Modifikationen betrachtet: So löst man sich zum einen von der baumartigen Beziehung zwischen Membranen und ermöglicht das Arbeiten auf einer Zellstruktur, die durch eine etwas mächtigere Klasse von gerichteten Graphen repräsentiert werden kann. Weiterer Aspekte sind die Beschränkung auf globale Evolutionsregeln, d. h. identische Regelmengen für jede Region, sowie der Verzicht auf Zielindikatoren, wobei die Resultate einer Splicing-Operation grundsätzlich in alle benachbarten Regionen übergehen. Wir werden hier keine dieser Varianten von Splicing-P-Systemen näher untersuchen, da die im bisherigen und weiteren Verlauf der Arbeit angegebenen Systeme ohne solche Modifikationen arbeiten, und verweisen für eine genauere Betrachtung auf die angegebene Quelle.

Stattdessen werden wir zwei Resultate zu Splicing-P-Systemen in ihrer hier eingeführten, ursprünglichen Form vorstellen. Dabei untersuchen wir den Einfluss der erlaubten Komplexität (maximaler Grad und Tiefe) der zugrunde liegenden Membranstruktur und der Länge der Kontexte bei Splicing-Regeln, also des Durchmessers, auf die Klasse der von solchen Systemen erzeugbaren Sprachen. Zuvor sei noch festgehalten, dass wegen der Äquivalenz einer Evolutionsregel $(u_1\#u_2\$u_3\#u_4, tar_1, tar_2)$ zu der Regel $(u_3\#u_4\$u_1\#u_2, tar_2, tar_1)$ folgt, dass

$$\mathcal{SPL}_m(d, p_1, p_2, p_3, p_4) = \mathcal{SPL}_m(d, p_3, p_4, p_1, p_2),$$

für $m, d \in \mathbb{N}^+, p_i \in \mathbb{N} \cup \{*\}, i \in [4]^+$. Offensichtlich gilt auch

$$\mathcal{SPL}_m(d, p_1, p_2, p_3, p_4) \subseteq \mathcal{SPL}_{m'}(d', p'_1, p'_2, p'_3, p'_4),$$

für $m \leq m', d \leq d', p_i \leq p'_i, i \in [4]^+$, wobei gilt $p \leq *$ für alle $p \in \mathbb{N} \cup \{*\}$.

Wir suchen also möglichst minimale Parameter, mit denen z. B. die Beschreibungsstärke von Typ-0-Grammatiken erreicht werden kann. Zunächst zeigen wir jedoch folgende Aussage ([Fri08], Theorem 9.1):

Theorem 5.1. $\mathcal{SPL}_1(1, 0, 1, 1, 1) = \mathcal{SPL}_1(1, 1, 1, 0, 1) = REG$. ■

Beweis. (i) $REG \subseteq \mathcal{SPL}_1(1,0,1,1,1)$.

Sei $G = (N, T, S, P)$ eine rechts-lineare, reguläre Grammatik und $W, Z \notin N \cup T$. Wir wollen nun ein Splicing-P-System Π angeben, welches G simuliert. Da G rechts-linear ist, tauchen Nichtterminalsymbole $X \in N$ in jeder Satzform, wenn überhaupt, nur am rechten Rand auf. Damit besteht keine Notwendigkeit, die Satzformen in Π rotieren zu lassen und auch die zusätzliche Kennzeichnung des linken und rechten Wortendes durch Zusatzsymbole kann entfallen.

Wir definieren $\Pi = (V, T, \mu, L_1, R_1)$ mit Grad 1, Tiefe 1 und mit Durchmesser $dia(\Pi) = (0,1,1,1)$ wie folgt:

$$V = N \cup T \cup \{W, Z\}$$

$$\mu = (\{1\}, \emptyset)$$

$$L_1 = \{S\} \cup \{ZaY \mid X \to aY \in P,\ X, Y \in N,\ a \in T\} \cup \{Wa \mid X \to a \in P,\ X \in N,\ a \in T\}$$

$$R_1 = \{(\varepsilon \# X \$ Z \# a; here, here) \mid X \to aY \in P,\ X, Y \in N,\ a \in T\} \cup \{(\varepsilon \# X \$ W \# a; out, here) \mid X \to a \in P,\ X \in N,\ a \in T\}$$

Π erzeugt in Region 1 ausgehend von S alle Satzformen βX von G, mit $\beta \in T^*$ und $X \in N$. Für ein beliebiges solches βX und eine Produktionsregel $r_1 = X \to aY$, $r_1 \in P$, $a \in T$, $Y \in N$, gilt nun $\beta X \Rightarrow_{r_1} \beta aY$, d. h. βaY ist ebenfalls eine Satzform von G. Π simuliert dies durch folgendes Splicing:

$$(\beta|X, Z|aY) \vdash_{(\varepsilon \# X \$ Z \# a; here, here)} (\underbrace{\beta aY}_{here}, \underbrace{ZX}_{here})$$

Sobald eine Regel der Form $X \to a$ simuliert wird, enthält die Satzform nur noch Terminalsymbole und wird an die Umgebung abgegeben:

$$(\beta|X, W|a) \vdash_{(\varepsilon \# X \$ W \# a; out, here)} (\underbrace{\beta a}_{out}, \underbrace{WX}_{here})$$

Da jede Splicing-Operation in Π der Anwendung einer Produktionsregel von G entspricht und das System von S ausgehend startet, gilt für alle $w \in T^*$:

$$w \in L(\Pi) \text{ gdw. } w \in L(G), \text{ und somit } L(\Pi) = L(G).$$

(ii) $REG \supseteq \mathcal{SPL}_1(1,0,1,1,1)$.

Auf die Angabe des zweiten Teils des Beweises verzichten wir in dieser Arbeit, da er einige weitere, im Rahmen der vorliegenden Arbeit nicht eingeführte Konzepte voraussetzt. Aus dem gleichen Grund wird er auch in [Fri08] nicht angegeben, er findet sich aber in [PY99]. □

In [Fri08] findet sich die Aussage (Theorem 9.2) $\mathcal{SPL}_2(2,1,2,2,1) = RE$, der Beweis ist dort jedoch fehlerhaft und scheint nicht ohne Weiteres unter Beibehaltung der ursprünglichen These zu korrigieren. In dieser Arbeit zeigen wir stattdessen das folgende Resultat:

Theorem 5.2. $\mathcal{SPL}_3(3,0,2,2,0) = \mathcal{SPL}_3(3,2,0,0,2) = RE$. ■

Beweis. (i) $RE \subseteq \mathcal{SPL}_3(3,0,2,2,0)$.[10]

Sei $G = (N,T,S,P)$ eine Typ-0-Grammatik in Kuroda-Normalform. Wir konstruieren ein Splicing-P-System Π, welches G simuliert. Dabei kommt erneut eine Variante von Rotate-and-Simulate zum Einsatz, wir kennzeichnen die Anfänge einer Satzform daher wie gehabt mit $B \notin N \cup T$. Zudem werden wir alle diese Symbole durchnummerieren, d. h. wir definieren eine bijektive Funktion $f\colon N \cup T \cup \{B\} \to [n]^+$, mit $|N \cup T \cup \{B\}| = n$ und schreiben kurz α_i für $\alpha \in N \cup T \cup \{B\}$ mit $f(\alpha) = i$, d. h. $N \cup T \cup \{B\} = \{\alpha_1, \ldots, \alpha_n\}$.

Da G in Kuroda-Normalform vorliegt, gilt für jede Produktionsregel $p = u \to v \in P$ entweder $|u| = 1$ oder $|u| = 2$. Dementsprechend wollen wir P in die beiden disjunkten Mengen P_1 und P_2 aufteilen, mit $P_k = \{u \to v \in P \mid |u| = k\}$, $k \in \{1,2\}$, $P_1 \cup P_2 = P$, $P_1 \cap P_2 = \emptyset$, $|P| = m$, $|P_1| = m_1$, $|P_2| = m_2$ und $m_1 + m_2 = m$. Weiterhin definieren wir die Menge $P' = \{\alpha_i \to \alpha_i \mid i \in [n]^+\}$, deren Erläuterung später folgt. Darüber hinaus verwenden wir die Hilfssymbole $o, X, X_1, X_2, Y, Y_1, Y_2, Z_X, Z_{X_1^+}, Z_{X_2^+}, Z_Y, Z_{Y_1^+}, Z_{Y_2^+}, Z_\varepsilon, Z_{X_i}, Z_{Y_i}, Y'_j, Z_{Y'_j}, Z_k \notin N \cup T$, mit $1 \leq i \leq n+m$, $k \in [n]^+$ und $n + m_1 < j \leq n + m$.

Alle Regeln $u \to v \in P' \cup P_1 \cup P_2$ werden ebenfalls nummeriert, wir definieren daher mit $g\colon P' \cup P \to [n+m]^+$ eine bijektive Funktion, so dass $g(p) > n + m_1$ für alle $p \in P_2$. Wir schreiben p_i für $p = u \to v \in P' \cup P$ mit $g(p) = i$. Die linke bzw. rechte Seite dieser Regel werden wir auch direkt mit u_i bzw. v_i adressieren. Bevor wir die formale Beschreibung von Π angeben, wollen wir einen Überblick über dessen Funktionsweise geben.

Eine Regelanwendung auf eine Satzform $w_1 u_i w_2$ von G, wobei $w_1, w_2 \in (N \cup T)^*$ und $u_i \to v_i \in P$ die angewandte Regel ist, wird in Π dadurch simuliert, dass wir bei einer Rotation $w_2 B w_1 u_i$ der Satzform u_i durch o^i ersetzen, somit also $w_2 B w_1 o^i$ erhalten. Zudem wird auf der linken Seite $o^j v_j$ konkateniert, so dass $o^j v_j w_2 B w_1 o^i$ vorliegt. Danach werden schrittweise die o's auf beiden Seiten entfernt, so dass wir nur im gewünschten Fall $i = j$ zu dem Resultat $v_i w_2 B w_1$ kommen. Man beachte, dass durch das Entfernen von u_i auf der rechten Seite und dem Anfügen von v_i auf der linken Seite mit jeder Regelanwendung gleichzeitig eine Rotation einhergeht.

In diesem Zusammenhang sind auch die Regeln $\alpha \to \alpha \in P'$, $\alpha \in N \cup T \cup \{B\}$ zu verstehen. Deren Repräsentationen dienen in Π dazu, durch eine Regelanwendung nach dem gleichen Prinzip reine Rotationsschritte vorzunehmen: aus $w\alpha$, $w \in (N \cup T \cup \{B\})^*$ wird somit αw.

[10] Der Beweis orientiert sich an der in [Fri08] vorgestellten Idee. Im Zuge der Korrektur des dort konstruierten Splicing-P-Systems wurde eine weitere Membran hinzugefügt. Im Gegenzug verfügt das hier konstruierte System über einen geringeren Durchmesser.

Die Satzformen und ihre Rotationen liegen in Π in der Form $w_2' B w_1'$ vor, mit $w_1', w_2' \in (N \cup T)^*$. Im Fall $w_1' = \varepsilon$, wenn also B am Wortende steht, ist w_2' eine gültige Satzform von G und wird von Π an die Umgebung abgegeben, der zusätzliche Rotationsschritt zu Bw_2' kann ausgelassen werden. Zusätzlich werden wir mit den Hilfssymbolen X, Y (und einigen Varianten davon, welche wir zum Speichern von Informationen über vorangegangene Splicing-Operationen verwenden) den linken und rechten Wortrand markieren.

Wir definieren das G simulierende Splicing-P-System $\Pi = (V, T, \mu, L_1, L_2, L_3, R_1, R_2, R_3)$ mit Grad 3, Tiefe 3 und Durchmesser $dia(\Pi) = (0, 2, 2, 0)$ wie folgt:

- $V = N \cup T \cup \{Z_{X_i}, Z_{Y_i} \mid i \in [n+m]^+\} \cup$
  $\{Y_i', Z_{Y_i'} \mid n + m_1 < i \leq n + m\} \cup \{Z_i, Z_i' \mid i \in [n]^+\} \cup$
  $\{B, o, X, X_k, Y, Y_k, Z_X, Z_{X_k^+}, Z_Y, Z_{Y_k^+}, Z_\varepsilon \mid k \in \{1, 2\}\}$
- $\mu = (\{1, 2, 3\}, \{(1, 2), (2, 3)\})$
- $L_1 = \{Z_i' \alpha_i Y \mid i \in [n]^+,\ \alpha_i \in N \cup T \cup \{B\}\} \cup \{Z_\varepsilon\}$
- $L_2 = \{XBSY\} \cup \{X_2 Z_{X_2^+}, Z_{Y_1^+} Y_1'\} \cup \{Z_{Y_i} o^i Y_1 \mid i \in [n+m]^+\} \cup$
  $\{Z_{Y_i'} Y_i' \mid n + m_1 < i \leq n + m\} \cup \{X \alpha_i Z_i \mid i \in [n]^+,\ \alpha_i \in N \cup T \cup \{B\}\}$
- $L_3 = \{Z_{Y_2^+} Y_2, X_1 Z_{X_1^+}\} \cup \{X_1 o^i v_i Z_{X_i} \mid i \in [n+m]^+\}$
- $R_1 = \{r_{1,i} = (\varepsilon \# \alpha_i Y_2 \$ Z_i' \# \varepsilon; in, out) \mid i \in [n]^+\} \cup$
  $\{r_2 = (\varepsilon \# Z_\varepsilon \$ X \# \varepsilon; here, out),$
  $r_3 = (\varepsilon \# B Y_2 \$ Z_\varepsilon \# \varepsilon; out, out)\}$
- $R_2 = \{r_{4,i} = (\varepsilon \# u_i Y \$ Z_{Y_i} \# \varepsilon; in, out) \mid i \in [n + m_1]^+\} \cup$
  $\{r_{5,i} = (\varepsilon \# u_2' Y \$ Z_{Y_i'} \# \varepsilon; here, out),$
  $r_{6,i} = (\varepsilon \# u_1' Y_i' \$ Z_{Y_i} \# \varepsilon; in, out) \mid u_i = u_1' u_2',\ u_1', u_2' \in N,$
  $n + m_1 < i \leq n + m,\ u_i \to v_i \in P_2\} \cup$
  $\{r_7 = (\varepsilon \# Z_{X_2^+} \$ X_1 \# \varepsilon; here, out),$
  $r_8 = (\varepsilon \# o Y_2 \$ Z_{Y_1^+} \# \varepsilon; in, out)\} \cup$
  $\{r_{9,i} = (\varepsilon \# Z_i \$ X_1 \alpha_i \# \varepsilon; out, out) \mid i \in [n]^+\}$
- $R_3 = \{r_{10,i} = (\varepsilon \# Z_{X_i} \$ X \# \varepsilon; here, out) \mid i \in [n+m]^+\} \cup$
  $\{r_{11} = (\varepsilon \# Y_1 \$ Z_{Y_2^+} \# \varepsilon; out, out),$
  $r_{12} = (\varepsilon \# Z_{X_1^+} \$ X_2 o \# \varepsilon; out, out)\}$

In Region 2 liegen nun Wörter der Form $X w u_i Y$, $w, u_i \in (N \cup T \cup \{B\})^*$ vor, in der initialen Konfiguration ist dies $XBSY$. Dabei gilt entweder $|u_i| = 1$ oder $|u_i| = 2$. In ersterem Fall wird eine Regel $p_i \in P \cup P'$, $i \in [n + m_1]^+$, simuliert. Dies beginnt mit dem Splicing

$$(Xw|u_iY, Z_{Y_i}|o^iY_1) \vdash_{r_{4,i}} (\underbrace{Xwo^iY_1}_{in}, \underbrace{Z_{Y_i}u_iY}_{out}).$$

Bei einer Simulation von $p_i \in P_2$, $n + m_1 < i \leq n + m$, ist dies in zwei Einzelschritte aufgeteilt, da der Durchmesser von Π die Regel $r_{4,i}$ für $|u_i| = 2$ nicht zulässt. Sei also $u_i = u'_1u'_2$, $u'_1, u'_2 \in N$, dann können folgende Evolutionsregeln angewendet werden:

$$(Xwu'_1|u'_2Y, Z_{Y'_i}|Y'_i) \vdash_{r_{5,i}} (\underbrace{Xwu'_1Y'_i}_{here}, \underbrace{Z_{Y'_i}u'_2Y}_{out}),$$

$$(Xw|u'_1Y'_i, Z_{Y_i}|o^iY_1) \vdash_{r_{6,i}} (\underbrace{Xwo^iY_1}_{in}, \underbrace{Z_{Y_i}u'_1Y'_i}_{out}).$$

Bei beiden Varianten wird u_iY am Wortende durch o^iY_1 ersetzt, und das daraus resultierende Wort Xwo^iY_1 in Region 3 geschickt. Die bei diesen Operationen an Region 1 abgegebenen Wörter können dort nicht weiter Gegenstand von Regelanwendungen werden.

In Region 3 können auf Xwo^iY_1 nun folgende Regeln zur Anwendung kommen:

$$(Xwo^i|Y_1, Z_{Y_2^+}|Y_2) \vdash_{r_{11}} (\underbrace{Xwo^iY_2}_{here}, \underbrace{Z_{Y_2^+}Y_1}_{out}),$$

$$(X_1o^jv_j|Z_{X_j}, X|wo^iY_2) \vdash_{r_{10,j}} (\underbrace{X_1o^jv_jwo^iY_2}_{out}, \underbrace{XZ_{X_j}}_{out}),\ j \in [n+m]^+.$$

Dabei gehen $Z_{Y_2^+}Y_1$, XZ_{X_j} und $X_1o^jv_jwo^iY_2$ in Region 2 über, wo die ersteren beiden Wörter nicht weiter Teil einer Splicing-Operation werden können.

In Region 3 kann $r_{10,j}$, $j \in [n + m]^+$, bereits vor r_{11} angewendet werden, d. h. auf ein Wort der Form Xwo^iY_1. Dies gehört nicht zum Verlauf der Simulation einer Regelanwendung und ist somit eigentlich ein unerwünschtes Verhalten, wir wollen daher zeigen, dass dies keine Auswirkungen auf $L(\Pi)$ hat:

$$(X_1o^jv_j|Z_{X_j}, X|wo^iY_1) \vdash_{r_{10,j}} (\underbrace{X_1o^jv_jwo^iY_1}_{out}, \underbrace{XZ_{X_j}}_{out}),\ j \in [n+m]^+.$$

XZ_{X_j} kann in Region 2 an keinem Splicing teilnehmen, auf $X_1o^jv_jwo^iY_1$ ist hingegen zusammen mit $X_2Z_{X_2^+}$ zunächst noch r_7 anwendbar:

$$(X_2|Z_{X_2^+}, X_1|o^jv_jwo^iY_1) \vdash_{r_7} (\underbrace{X_2o^jv_jwo^iY_1}_{here}, \underbrace{X_1Z_{X_2^+}}_{out}).$$

$X_1Z_{X_2^+}$ geht in Region 1 über, wo es an keinen weiteren Splicing-Operationen teilnimmt. Das erste Resultat bleibt in der Region, darauf sind aber ebenfalls keine weiteren Regeln anwendbar.

Allgemein wollen wir festhalten, dass wir in Π bei sämtlichen Splicing-Operationen $(x, y) \vdash (z, w)$ nur dem Resultat z eine Bedeutung beimessen, wohingegen w immer als Nebenprodukt angesehen wird. w besteht fast ausschließlich aus den verwendeten Hilfssymbolen und wandert nach Definition unserer Regeln grundsätzlich eine Membranebene nach außen. Falls es dabei aus Region 1 an die Umgebung abgegeben wird, bleibt dies ohne Einfluss auf die erzeugte Sprache $L(\Pi)$, da immer Symbole $x \notin T$ enthalten sind. Wandert es in Region 1 oder 2, kann es dort nicht mehr Teil einer weiteren Regelanwendung werden. Wir werden diese Betrachtung aus Gründen der Übersicht bei der Angabe weiterer Splicing-Regeln nicht mehr ausführen, der/die geneigte Leser/in sei aber ermuntert, dies im Einzelfall trotzdem nachzuvollziehen.

Kehren wir nun zurück zur Betrachtung von $X_1o^jv_jwo^iY_2$, welches aus Region 3 nach Anwendung von $r_{10,j}$ nach außen abgegeben wurde. Das System wird nun abwechselnd immer rechts und links ein o entfernen, um die Wörter herauszufiltern, bei denen eine korrekte Ersetzung von u_i durch v_i stattgefunden hat. Dies beginnt in Region 2 durch die Anwendung folgender Evolutionsregeln:

$$(X_2|Z_{X_2^+}, X_1|o^jv_jwo^iY_2) \vdash_{r_7} (\underbrace{X_2o^jv_jwo^iY_2}_{here}, \underbrace{X_1Z_{X_2^+}}_{out}),$$

$$(X_2o^jv_jwo^{i-1}|oY_2, Z_{Y_1^+}|Y_1) \vdash_{r_8} (\underbrace{X_2o^jv_jwo^{i-1}Y_1}_{in}, \underbrace{Z_{Y_1^+}oY_2}_{out}).$$

Hierbei wird X_1 durch X_2 ersetzt sowie oY_2 durch Y_1. Das Ergebnis $X_2o^jv_jwo^{i-1}Y_1$ wandert in Region 3, die an Region 1 abgegebenen Wörter bleiben wiederum ohne Einfluss auf die weitere Berechnung.

Beim eben betrachteten Schritt ist die Evolutionsregel r_8 auch vor r_7 anwendbar, d. h. unter Beteiligung von $X_1o^jv_jwo^iY_2$, wobei oY_2 durch Y_1 ersetzt wird. Daraus resultiert das Wort $X_1o^jv_jwo^{i-1}Y_1$, welches in Region 3 wandert. Dort wird mittels einer Anwendung von r_{11} wie bereits betrachtet Y_1 durch Y_2 ersetzt, auf $X_1o^jv_jwo^{i-1}Y_2$ ist in Region 3 dann allerdings keine weitere Regel anwendbar.

Wir wenden uns wieder der ursprünglichen Berechnung zu, bei welcher nun in Region 3 ein Vorkommen von $X_2o^jv_jwo^{i-1}Y_1$ weitere Splicing-Operationen ermöglicht. Dabei wird Y_1 durch Y_2 ersetzt. Außerdem wird durch eine weitere Regel X_2o zu X_1 und das Resultat $X_1o^{j-1}v_jwo^{i-1}Y_2$ wieder in Region 2 geschickt:

$$(X_2o^jv_jwo^{i-1}|Y_1, Z_{Y_2^+}|Y_2) \vdash_{r_{11}} (\underbrace{X_2o^jv_jwo^{i-1}Y_2}_{here}, \underbrace{Z_{Y_2^+}Y_1}_{out}),$$

$$(X_1|Z_{X_1^+}, X_2o|o^{j-1}v_jwo^{i-1}Y_2) \vdash_{r_{12}} (\underbrace{X_1o^{j-1}v_jwo^{i-1}Y_2}_{out}, \underbrace{X_2oZ_{X_1^+}}_{out}).$$

Die Evolutionsregel r_8 entfernt in Region 2 immer am Wortende ein o, wohingegen dies in Region 3 durch r_{12} am Wortanfang geschieht. Beide Regeln schicken ihr Ergebnis in die Region der jeweils anderen Regel. Dabei muss eine Anwendung von r_8 in Region 2 immer durch eine Anwendung von r_{11} in Region 3 vorbereitet werden, bei welcher Y_1 durch Y_2 ersetzt wird. Analog liegt der „gespiegelte“ Fall bei den Regeln r_{12} und r_7 vor.

Das Entfernen der o's am Wortende und Wortanfang vollzieht sich also im Wechsel zwischen Region 3 und 2. Ausgehend von $X_1 o^j v_j w o^i Y_2$ unterscheiden wir dabei folgende Fälle:

$i > j$: Dann liegt nach j-facher Anwendung von r_8 und r_{12} in Region 2 das Wort $X_1 v_j w o^{i'} Y_2$ vor, für $i' = i - j > 0$, $v_j = \alpha_k v'$, $\alpha_k \in (N \cup T \cup \{B\})$ und $v' \in N \cup \{\varepsilon\}$.

Die Evolutionsregel r_8 kann weiterhin ein o am rechten Ende entfernen und das Wort $X_1 v_j w o^{i'-1} Y_1$ an Region 3 senden. Falls zuerst noch r_7 angewendet wird und X_1 durch X_2 ersetzt wird, wandert als Resultat von r_8 entsprechend $X_2 v_j w o^{i'-1} Y_1$ nach 3. In beiden Fällen kann in Region 3 nur noch Regel r_{11} zur Anwendung kommen. Deren jeweilige Resultate $X_1 v_j w o^{i'-1} Y_2$ bzw. $X_2 v_j w o^{i'-1} Y_2$ verbleiben danach ohne weitere Beteiligung an Splicing-Operationen in der Region.

In Region 2 kann auch folgende Regelanwendung stattfinden:

$$(X\alpha_k|Z_k, X_1\alpha_k|v' w o^{i'} Y_2) \vdash_{r_{9,k}} (\underbrace{X v_j w o^{i'} Y_2}_{out}, \underbrace{X_1 \alpha_k Z_k}_{out}).$$

In Region 1 ist auf $X v_j w o^{i'} Y_2$ jedoch nur noch r_2 anwendbar, welche das Präfix X entfernt. Sowohl $X v_j w o^{i'} Y_2$ als auch $v_j w o^{i'} Y_2$ bleiben in Region 1, wo keine neuen Regelanwendungen mehr darauf möglich sind.

$i < j$: Dann liegt nach i-facher Anwendung von r_8 und r_{12} in Region 2 das Wort $X_1 o^{j'} v_j w Y_2$ vor, für $j' = j - i$.

Darauf kann nur r_7 zur Anwendung kommen. Das Resultat $X_2 o^{j'} v_j w Y_2$ kann an keinen weiteren Splicing-Operationen teilnehmen.

$i = j$: In diesem Fall ist u_i korrekt durch v_i ersetzt worden. Es liegt dann nach i-facher Anwendung der Regeln r_8 und r_{12} in Region 2 das Wort $X_1 v_i w Y_2$ vor, mit $v_i = \alpha_k v'$, $\alpha_k \in (N \cup T \cup \{B\})$ und $v' \in N \cup \{\varepsilon\}$.

Eine Anwendung von r_7 liefert $X_2 v_i w Y_2$ und bleibt ohne weiteren Effekt. Hingegen liefert das Splicing

$$(X\alpha_k|Z_k, X_1\alpha_k|v' w Y_2) \vdash_{r_{9,k}} (\underbrace{X v_i w Y_2}_{out}, \underbrace{X_1 \alpha_k Z_k}_{out})$$

ein Vorkommen von $X v_i w Y_2$ in Region 1.

Sei $v_i w = w' \alpha_{k'}$ für $\alpha_{k'} \in N \cup T \cup \{B\}$ und $w' \in (N \cup T \cup \{B\})^*$. Dann liefert die Anwendung der Evolutionsregel $r_{1,k'}$

$$(Xw'|\alpha_{k'}Y_2, Z'_{k'}|\alpha_{k'}Y) \vdash_{r_{1,k'}} (\underbrace{Xv_iwY}_{in}, \underbrace{Z'_{k'}\alpha_{k'}Y_2}_{out})$$

ein Vorkommen von Xv_iwY in Region 2. Dort waren wir ursprünglich von dem Wort Xwu_iY ausgegangen, somit hat das System eine erfolgreiche Regelanwendung mit Rotation (bzw. reine Rotation, falls $p_i \in P'$) vorgenommen und kann nun auf Xv_iwY weitere Regelanwendungen simulieren.

Eine Anwendung von r_2 liefert

$$(\varepsilon|Z_\varepsilon, X|v_iwY_2) \vdash_{r_2} (\underbrace{v_iwY}_{here}, \underbrace{XZ_\varepsilon}_{out}).$$

Daraufhin kann $r_{1,k'}$ analog zum zuvor betrachteten Splicing das Wort v_iwY in Region 2 senden. Dort wird es in der anfangs betrachteten Art und Weise in der Form $w''o^{i'}Y_1$ ($v_iw = w''u_{i'}$, $p_{i'} \in P' \cup P$) in Region 3 geschickt, wo keine weiteren Regelanwengungen mehr erfolgen können.

Falls $\alpha_{k'} = B$, d. h. $Xv_iwY_2 = Xw'BY_2$, dann ist w' eine Satzform von G. In diesem Fall kommt, nachdem wir mittels r_2 das X abgeschnitten haben, auf $w'BY_2$ die Evolutionsregel r_3 wie folgt zum Einsatz:

$$(w'|BY_2, Z_\varepsilon|\varepsilon) \vdash_{r_3} (\underbrace{w'}_{out}, \underbrace{Z_\varepsilon BY_2}_{out}).$$

Die Satzform w' wird dabei an die Umgebung abgegeben.

Π generiert also von $XSBY$ ausgehend alle Satzformen w' von G, und gibt diese an die Umgebung ab. Falls $w' \in L(G)$, dann gilt $w' \in T^*$ und somit $w' \in L(\Pi)$. Andere Wörter sind in $L(\Pi)$ nicht enthalten. Somit folgt $L(G) = L(\Pi)$.

(ii) $\mathcal{SPL}_3(3, 0, 2, 2, 0) \subseteq RE$.

Wir wollen hier keinen formalen Beweis der zweiten Aussage geben, [Fri08] lässt eine entsprechende Betrachtung ebenfalls aus. Da jedes Splicing-P-System Π mit $L(\Pi) \in \mathcal{SPL}_3(3, 0, 2, 2, 0)$ ein diskretes Rechenmodell darstellt, welches auf endlichen Sprachen mittels Konkatenation von Teilwörtern neue Wörter erzeugt, vermuten wir allerdings nicht, damit den Rahmen der Turing-Berechenbarkeit zu verlassen, sondern erwarten, dass Π durch eine entsprechende Turingmaschine M_Π simuliert werden kann. Auf die formale Angabe einer solchen Konstruktion wollen wir aber aus Gründen der Komplexität verzichten. □

6 Anwendung am Beispiel des Hamiltonkreisproblems

6.1 Das Hamiltonkreisproblem

Sei $G = (Q, E)$ ein gerichteter Graph mit der Knotenmenge $Q = \{q_1, \ldots, q_n\}$, $n \in \mathbb{N}$ und den Kanten $E \subseteq Q \times Q$.

Ein *Zyklus* in G ist ein Pfad $v_1, \ldots, v_m$ ($v_i \in Q$, $i \in [m]^+$, $m > 1$), wobei $v_1 = v_m$. Ein *Kreis* ist ein Zyklus $v'_1, \ldots, v'_l$ ($v'_i \in Q$, $i \in [l]^+$, $l > 1$), bei dem nur Start- und Endknoten gleich sind, es gilt also zusätzlich $v'_i \neq v'_j$ für alle $i, j \in [l-1]^+$, $i \neq j$. Ein Kreis $v''_1, \ldots, v''_k$ von G ($v''_i \in Q$, $i \in [k]^+$, $k > 1$) heißt *Hamiltonkreis*, wenn er jeden Knoten $q \in Q$ genau einmal enthält (mit Ausnahme des Start und Endknotens), d. h. wenn $\{v''_1, \ldots, v''_{k-1}\} = Q$.

Das (gerichtete) *Hamiltonkreisproblem* ist die Frage, ob ein solcher Graph G einen Hamiltonkreis enthält, und gehört zur Klasse der NP-vollständigen Probleme. Als Problemgröße betrachten wir hierbei die Anzahl der Knoten des Graphen $n = |Q|$. Wir werden im nächsten Abschnitt zeigen, wie sich zu jedem Graphen ein Splicing-P-System konstruieren lässt, das dessen Hamiltonkreise ausgibt.

6.2 Konstruktion des Splicing-P-Systems

Sei $G_H = (Q, E)$ ein beliebiger, gerichteter Graph mit $Q = \{q_1, \ldots, q_n\}$, $E \subseteq Q \times Q$, $n \geq 2$. Wir konstruieren dazu das Splicing-P-System Π_H, welches die Frage beantwortet, ob G_H einen Hamiltonkreis enthält. Falls $L(\Pi_H) = \emptyset$ so besitzt G_H keinen Hamiltonkreis, anderenfalls enthält $L(\Pi_H)$ alle Hamiltonkreise des Graphen.

Bei der Beschreibung eines solchen Pfades $v_1, \ldots, v_n, v_1$ ($v_i \in Q$, $i \in [n]^+$, $n = |Q|$) werden wir auf die Angabe des Endknotens verzichten. Einen Knoten $q_i \in Q$, $i \in [n]^+$, werden wir außerdem einfach durch i repräsentieren und dabei auch die Kommata nicht mit notieren. Der zyklische Pfad q_1, q_3, q_2, q_1 wird demnach mittels der Ziffernfolge 132 dargestellt. Die zyklischen Permutationen dieser Folge werden wir dabei als äquivalent betrachten. Wenn $v_1, \ldots, v_n$ ein Hamiltonkreis ist, dann gilt dies offensichtlich auch für $v_n, v_1, \ldots, v_{n-1}$. Beide Varianten beschreiben – die Kennzeichnung eines bestimmten Start- und Endknotens einmal außer Acht gelassen – den gleichen Zyklus, daher soll auch Π_H zu jeder Menge in dieser Hinsicht äquivalenter Kreise nur einen Repräsentanten ausgeben.

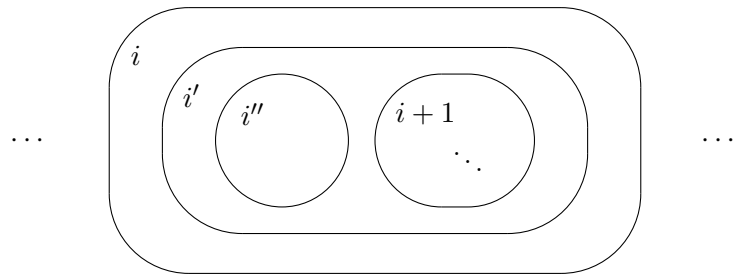

Abbildung 5: Ausschnitt aus der Membranstruktur von Π_H (für $i \in [n]^+$).

Wir definieren das System Π_H mit Grad $3n + 2$, Tiefe $2n + 2$ und Durchmesser $dia(\Pi_H) = (1, 2, 2, 1)$ wie folgt:

$$\Pi_H = (V, T, \mu, L_{1_O}, L_1, L_{1'}, L_{1''}, L_2, L_{2'}, L_{2''}, \ldots, L_n, L_{n'}, L_{n''}, L_{n+1}, R_{1_O}, R_1, R_{1'}, R_{1''}, \ldots, R_n, R_{n'}, R_{n''}, R_{n+1}), \text{ wobei}$$

- $V = \{i, \underline{i}, X_{i,j}, Y_{i,j}, Z_{X_{i,j}}, Z_{Y_{i,j}}, Z_{\underline{i}} \mid i, j \in [n]^+\} \cup \{X, Y, Z_X, Z_Y, Z_\varepsilon\}$,
- $T = [n]^+$,
- $\mu = (Q_\mu, E_\mu)$ mit
  $Q_\mu = \{1_O, n + 1\} \cup \{i, i', i'' \mid i \in T\}$[11] und
  $E_\mu = \{(1_O, 1)\} \cup \{(i, i'), (i', i''), (i', i + 1) \mid i \in T\}$,
- $L_{1_O} = \{Z_\varepsilon\}$,
- $L_i = \{Z_Y Y, Z_{\underline{i}} i Y\}$,
- $L_{i'} = \{X Z_X\} \cup \{Z_{Y_{i,j}} Y_{i,j} \mid j \in T\}$,
- $L_{i''} = \{X_{i,j} \sigma_i(j) Z_{X_{i,j}}, X_{i,j} Z_{X_{i,j}} \mid j \in T\}$,
- $L_{n+1} = \{X \underline{j} Y \mid j \in T\}$,
- $R_{1_O} = \{r_{1_O,1} = (\varepsilon \# Z_\varepsilon \$ X \# \varepsilon; here, out),$
  $r_{1_O,2} = (\varepsilon \# Y \$ Z_\varepsilon \# \varepsilon; out, out)\}$,
- $R_i = \{r_{i,1} = (\varepsilon \# Y_{i,1} \$ Z_Y \# Y; in, out)\} \cup$
  $\{r_{i,2,j} = (\sigma_i(j) \# \underline{i} Y_{i,1} \$ Z_{\underline{i}} \# i; out, out) \mid (j, i) \in E\}$,
- $R_{i'} = \{r_{i',1,j} = (\varepsilon \# \sigma_i(j) Y \$ Z_{Y_{i,j}} \# Y_{i,j}; in, in) \mid j \in T\} \cup$
  $\{r_{i',2,j} = (\varepsilon \# Y_{i,j+1} \# Z_{Y_{i,j}} \# Y_{i,j}; in, in) \mid j \in [n-1]^+\} \cup$
  $\{r_{i',3} = (X \# Z_X \$ X_{i,1} \# \varepsilon; out, out)\}$,
- $R_{i''} = \{r_{i'',1,j} = (\sigma_i(j) \# Z_{X_{i,j}} \$ X \# \varepsilon; out, out) \mid j \in T\} \cup$
  $\{r_{i'',2,j} = (X_{i,j} \# Z_{X_{i,j}} \$ X_{i,j+1} \# \varepsilon; out, out) \mid j \in [n-1]^+\}$

  wobei $\sigma_i \colon T \to T \cup \underline{T}$ mit $\sigma_i(j) = \begin{cases} j & \text{falls } j > i, \\ \underline{j} & \text{sonst.} \end{cases}$

  } für alle $i \in T$

- $R_{n+1} = \{r_{n+1,1} = (\varepsilon \# Y \$ X \# \varepsilon; here, here),$
  $r_{n+1,2} = (\varepsilon \# Y \$ X \# \varepsilon; out, out)\}$.

Wir definieren zudem $\underline{T} = \{\underline{j} \mid j \in T\}$ und $\Sigma_i = \{\sigma_i(j) \mid j \in T\}$, $i \in T$.

[11] Bei unserer Beschreibung erlauben wir uns aus Gründen der Übersicht, die Menge der Membranen $\{1, \ldots, 3n + 2\}$ in der hier auftretenden Form zu repräsentieren.

Die Berechnung von Π_H läuft nach folgendem Prinzip ab:

(i) In Region $n+1$ werden zunächst alle möglichen „potentiellen“ Pfade erzeugt, d. h. alle Wörter $X\underline{i_1}\dots\underline{i_l}Y$ mit $i_j \in T$, $j \in [l]^+$, $l \in \mathbb{N}$, welche den Pfad $i_1 \dots i_l$ repräsentieren. Ob es sich dabei tatsächlich um einen Pfad von G_H handelt, also $(i_j, i_{j+1}) \in E$ für $j \in [l-1]^+$ gilt, spielt hierbei keine Rolle. Diese Überprüfung findet schrittweise zu späteren Zeitpunkten statt. Wir werden dann, falls $(i_{l-1}, i_l) \in E$, das $\underline{i_l}$ durch i_l ersetzen. Der Unterstrich kennzeichnet also eine noch nicht geprüfte Kante zwischen i_{l-1} und i_l.

In der initialen Konfiguration sind in Region $n+1$ alle Knoten $q_i \in Q$ in der Form $X\underline{i}Y$ vorhanden. Durch die Regel $r_{n+1,1}$ werden diese in allen Kombinationen aneinandergefügt. Somit sind nach einer Transition in Region $n+1$ zusätzlich alle Kombinationen der Länge 2 vorhanden und nach einer weiteren Transition alle Pfade mit maximaler Länge 4.

1. $(X\underline{i}|Y, X|\underline{j}Y) \vdash_{r_{n+1,1}} (X\underline{i}\,\underline{j}Y, XY)$,
2. $(X\underline{i}\,\underline{j}|Y, X|\underline{k}Y) \vdash_{r_{n+1,1}} (X\underline{i}\,\underline{j}\,\underline{k}Y, XY)$,
   $(X\underline{i}\,\underline{j}|Y, X|\underline{k}\,\underline{l}Y) \vdash_{r_{n+1,1}} (X\underline{i}\,\underline{j}\,\underline{k}\,\underline{l}Y, XY)$,
3. ... für $i, j, k, l \in T$.

Alle Pfade w mit $w \in \underline{T}^*$, $|w| = l$, $l \in [k]^+$, $k \in \mathbb{N}^+$, liegen nach $\lceil \log_2 k \rceil$ Transitionen in der Form XwY vor.

Die Regel $r_{n+1,2}$ erzeugt die gleichen Wörter XwY und schickt diese in Region n'. Das in beiden Regeln beim Splicing ebenfalls erzeugte Wort XY nimmt in Region n' an keiner weiteren Splicing-Operation teil.

(ii) In Region i', für $i \in T$, liegen im Verlaufe der Berechnung Wörter der Form XwY vor, mit $w \in (T \cup \underline{T})^*$. Dabei können wir uns die Regionen i' und i'' als ein Teilsystem ähnlich dem in Beispiel 5.2 betrachteten Splicing-P-System Π_2 vorstellen, welches alle Rotationen $Xw'Y$ mit $w' \in \mathit{cycle}(w)$ erzeugt. Da wir die Wörter hierbei als zyklische Pfade verstehen, kennzeichnen wir den Anfang von w nicht, alle w' beschreiben den gleichen Zyklus. Ein Rotationsschritt läuft nach folgendem Prinzip ab:

Sei $(M_1, M_{1'}, M_{1''}, \dots, M_i, M_{i'}, M_{i''}, \dots, M_{n''}, M_{n+1})$ eine Konfiguration von Π_H mit $XvaY \in M_{i'}$, $v \in (T \cup \underline{T})^*$, $a \in \Sigma_i$ und $a' = \sigma_i^{-1}(a)$. Dann führt das Splicing

$$(Xv|aY, Z_{Y_{i,a'}}|Y_{i,a'}) \vdash_{r_{i',1,a'}} (\underbrace{XvY_{i,a'}}_{in}, \underbrace{Z_{Y_{i,a'}}aY}_{in})$$

zu einer Ersetzung von aY durch $Y_{i,a'}$ am rechten Ende, wobei das Wort $XvY_{i,a'}$ in Region i'' gesandt wird. Dorthin wandert auch das Nebenprodukt $Z_{Y_{i,a'}}aY$, welches allerdings an keinem Splicing mehr teilnimmt. Beide Resultate gehen überdies in

Region $i+1$ über, wo ebenfalls keine weitere Regelanwendung möglich ist. Wir werden den letzteren Fall vorerst ausblenden und später noch einmal darauf zurückkommen.

In Region i'' wird im Wort $XvY_{i,a'}$ nun X durch $X_{i,j}\sigma_i(j)$ für alle $j \in T$ ersetzt:

$$(X_{i,j}\sigma_i(j)|Z_{X_{i,j}}, X|vY_{i,a'}) \vdash_{r_{i'',1,j}} (\underbrace{(X_{i,j}\sigma_i(j)vY_{i,a'}}_{out}, \underbrace{XZ_{X_{i,j}}}_{out}).$$

Hierbei messen wir dem zweiten Resultat $XZ_{X_{i,j}}$ erneut keine Bedeutung zu, es wandert in Region i', wo keine Regeln darauf zur Anwendung gebracht werden können.

Die Regeln $r_{i',2,a'-1}$ und $r_{i'',2,j-1}$ ersetzen ausgehend vom in i' nun vorliegenden $X_{i,j}\sigma_i(j)vY_{i,a'}$ die Randsymbole $Y_{i,a'}$ bzw. $X_{i,j}$ durch Exemplare mit einem um eins verringerten Index ($Y_{i,a'-1}$ und $X_{i,j-1}$, falls $a', j > 1$):

$$i': (X_{i,j}\sigma_i(j)v|Y_{i,a'}, Z_{Y_{i,a'-1}}|Y_{i,a'-1}) \vdash_{r_{i',2,a'-1}} (\underbrace{X_{i,j}\sigma_i(j)vY_{i,a'-1}}_{in}, \underbrace{Z_{Y_{i,a'-1}}Y_{i,a'}}_{in}),$$

$$i'': (X_{i,j-1}|Z_{X_{i,j-1}}, X_{i,j}|\sigma_i(j)vY_{i,a'-1}) \vdash_{r_{i'',2,j-1}} (\underbrace{X_{i,j-1}\sigma_i(j)vY_{i,a'-1}}_{out}, \underbrace{X_{i,j}Z_{X_{i,j-1}}}_{out}).$$

Die hierbei anfallenden Wörter $Z_{Y_{i,a'-1}}Y_{i,a'}$ und $X_{i,j}Z_{X_{i,j-1}}$ wandern jeweils in die andere Region und werden nicht weiter Teil von Regelanwendungen.

Das abwechselnde Dekrementieren wiederholt sich in dieser Art, bis mindestens einer der Indizes 1 ist. Wir unterscheiden hierbei – von $X_{i,j}\sigma(j)vY_{i,a'}$ ausgehend – zwischen drei Fällen:

$j > a'$: Dann liegt nach je $(a'-1)$-facher Anwendung einer der Regeln $r_{i',2,a'-z}$ abwechselnd mit $r_{i'',2,j-z}$ ($z \in \{1,\ldots,a'-1\}$) in Region i' das Wort $X_{i,j+1-a'}\sigma_i(j)vY_{i,1}$ vor. Auf dieses können keine weiteren Regeln angewendet werden.

$j < a'$: In diesem Fall liegt nach $(j-1)$-facher Anwendung je einer der Regeln $r_{i',2,a'-z}$ und $r_{i'',2,j-z}$ ($z \in \{1,\ldots,j-1\}$) in Region i' das Wort $X_{i,1}\sigma_i(j)vY_{i,a'+1-j}$ vor.

Dann führt einerseits $r_{i',2,a'-j}$ zu dem Ergebnis

$$(X_{i,1}\sigma_i(j)v|Y_{i,a'+1-j}, Z_{Y_{i,a'-j}}|Y_{i,a'-j}) \vdash_{r_{i',2,a'-j}} (\underbrace{X_{i,1}\sigma_i(j)vY_{i,a'-j}}_{in}, \underbrace{Z_{Y_{i,a'-j}}Y_{i,a'+1-j}}_{in}),$$

wonach auf $X_{i,1}\sigma_i(j)vY_{i,a'-j}$ in Region i'' keine Evolutionsregel mehr anwendbar ist.

Andererseits ergibt die Anwendung von $r_{i',3}$:

$$(X|Z_X, X_{i,1}|\sigma_i(j)vY_{i,a'+1-j}) \vdash_{r_{i',3}} (\underbrace{X\sigma_i(j)vY_{i,a'+1-j}}_{out}, \underbrace{X_{i,1}Z_X}_{out}).$$

Weitere Anwendungen auf $X\sigma_i(j)vY_{i,a'+1-j}$ finden in Region i nicht statt, da es nicht über das Suffix $Y_{i,1}$ verfügt.

$j = a'$: In diesem Fall gilt $\sigma_i(j) = a$ und es liegt nach $(j-1)$-facher Anwendung je einer der Regeln $r_{i',2,j-z}$ und $r_{i'',2,j-z}$ $(z \in \{1, \ldots, j-1\})$ in Region i' das Wort $X_{i,1}avY_{i,1}$ vor.

Darauf kommt die Evolutionsregel $r_{i',3}$ zur Anwendung:

$$(X|Z_X, X_{i,1}|avY_{i,1}) \vdash_{r_{i',3}} (\underbrace{XavY_{i,1}}_{out}, \underbrace{X_{i,1}Z_X}_{out}).$$

Das Wort $XavY_{i,1}$ wandert nach außen in Region i. Dort wird das Endsymbol $Y_{i,1}$ mittels der Regel $r_{i,1}$ durch Y ersetzt und das Resultat $XavY$ zurück nach i' geschickt:

$$(Xav|Y_{i,1}, Z_Y|Y) \vdash_{r_{i,1}} (\underbrace{XavY}_{in}, \underbrace{Z_YY_{i,1}}_{in}).$$

Dann ist die Rotation von $XvaY$ zu $XavY$ abgeschlossen.

(iii) Bei der zuletzt betrachteten Regelanwendung wollen wir uns bewusst machen, dass für alle Rotationen $w' \in cycle(w)$, $w \in (T \cup \underline{T})^*$, die in der Form $Xw'Y$ in Region i' erzeugt werden, eine Transition zuvor das Wort $Xw'Y_{i,1}$ in Region i vorliegt. Falls nun $w' = v'\sigma_i(j)\underline{i}$, $v' \in (T \cup \underline{T})^*$, kann die Regel $r_{i,2,j}$ in der Art

$$(Xv'\sigma_i(j)|\underline{i}Y_{i,1}, Z_{\underline{i}}|iY) \vdash_{r_{1,2,j}} (\underbrace{Xv'\sigma_i(j)iY}_{out}, \underbrace{Z_{\underline{i}}\underline{i}Y_{i,1}}_{out})$$

genau dann zur Anwendung kommen, wenn $(j, i) \in E$. $Xv'\sigma_i(j)iY$ wandert nach außen in Region $(i-1)'$. Dort beginnt dann der in (ii) beschriebene Ablauf von vorn.

Die in Region $n+1$ erzeugten Pfade XwY, $w \in \underline{T}^*$, werden also durch n', n'' und n nur dann in Region $(n-1)'$ weitergeleitet, wenn sie den Knoten $\underline{n}$ enthalten und dieser vor einem Knoten $\underline{j}$ steht, so dass $(j, n) \in E$. Dabei wird genau ein Vorkommen von $\underline{n}$ durch n ersetzt und der so entstandene Pfad Xw_nY nach außen abgegeben. Sollte w_n noch weitere Vorkommen von $\underline{n}$ enthalten, kann es in $(n-1)'$ nicht mehr vollständig rotiert werden. Die Regel $r_{(n-1)',2,j}$ ist nur auf Wörter mit dem Suffix $\sigma_{n-1}(j)Y$ anwendbar. Es gilt $\underline{n} \notin \Sigma_{n-1}$, daher endet die Bearbeitung eines solchen Wortes, sobald $\underline{n}$ am Ende steht.

Allgemein wird durch die Regionen i', i'' und i $(i \in T)$ in den Pfaden genau ein Vorkommen von $\underline{i}$ gesucht, durch i ersetzt und nach außen abgegeben, wenn $\underline{i}$ nach einem $\sigma_i(j)$ steht und dies einer Kante $(j, i) \in E$ entspricht. Sollte dabei der Knoten $\underline{k}$ am Ende auftauchen für $k > i$, dann muss der ursprüngliche Pfad mehrere Vorkommen von $\underline{k}$ enthalten haben. Es gilt $\underline{k} \notin \Sigma_i$ und die Bearbeitung des

Wortes endet. Dies ist durchaus gewollt, da ein solches Wort keinen Hamiltonkreis beschreiben kann.

Es ist zu beachten, dass eine Regelanwendung in Region i' ($i \in T$) mit Zielindikator in das entsprechende Resultat nicht nur an Region i'' sondern auch an $i+1$ schickt. Während ersteres dem Rotationsvorgang dient, ist letzteres Verhalten aus unserer Sicht eigentlich unerwünscht, da die Arbeitsweise des Systems dem Prinzip folgt, die in der elementaren Membran $n+1$ erzeugten Pfade stets schrittweise nach außen zu schicken, wobei ungültige Pfade aussortiert werden. Bei einem solchen Schritt können wir die Membranen i, i' und i'' als eine logische Funktionseinheit F_i ansehen, deren Arbeitsweise wir gerade betrachtet haben. Eine Kommunikation zwischen diesen Funktionseinheiten ist dabei von uns nur nach außen gerichtet beabsichtigt, d. h. in der Form der von F_i erzeugten Resultate, die an die Funktionseinheit F_{i-1} abgegeben werden. Um unerwünschte Effekte durch die Kommunikation in die andere Richtung zu vermeiden, verwendet jede Funktionseinheit F_i ihren eigenen Satz an Hilfssymbolen $X_{i,j}$, $Y_{i,j}$, $Z_{X_{i,j}}$, $Z_{Y_{i,j}}$, $Z_{\underline{i}}$ (für alle $j \in T$). Wenn nun aus Region i' Wörter nach Region $i+1$ wandern, so geschieht dies durch eine der Regeln $r_{i',1,j}$ oder $r_{i',2,j}$. Deren Resultate enthalten dann an mindestens einem Wortende eines der zu F_i gehörenden Hilfssymbole, so dass eine weitere Regelanwendung in $i+1$ nicht möglich ist.

(iv) Wenn ein Wort XwY, $w \in (T \cup \underline{T})^*$, von Region 1 an die Oberflächenmembran 1_O abgegeben wird, so gilt:

1. w enthält genau ein Vorkommen des Symbols i für alle $i \in T$, d. h. $|w| \geq n$.
2. Falls $|w| > n$, gilt demnach $w \notin T^*$.
3. Falls $|w| = n$, dann ist $w = i_1 \dots i_n$ und es gilt $\{i_1, \dots, i_n\} = T$. Zudem gilt $(i_j, i_{j+1}) \in E$ und $(i_n, i_1) \in E$ für alle $j \in [n-1]^+$. w beschreibt somit einen Hamiltonkreis des Graphen G_H.

Durch die Regeln $r_{1_O,1}$ und $r_{1_O,2}$ wird w letztendlich an die Umgebung abgegeben:

$$(\varepsilon | Z_\varepsilon, X | wY) \vdash_{r_{1_O,1}} (\underbrace{wY}_{here}, \underbrace{XZ_\varepsilon}_{out}),$$

$$(w | Y, Z_\varepsilon | \varepsilon) \vdash_{r_{1_O,2}} (\underbrace{w}_{out}, \underbrace{Z_\varepsilon Y}_{out}).$$

Wenn $w \in L(\Pi_H)$, dann muss wegen $w \in T^*$ der zuletzt betrachtete Fall (3.) gelten. Dann beschreibt w einen Hamiltonkreis von G_H. Da Π_H alle möglichen Pfade testet, gilt auch die umgekehrte Richtung: Wenn w einen Hamiltonkreis beschreibt, dann ist $w \in L(\Pi_H)$.

Für das Hamiltonkreisproblem ergibt sich: G_H besitzt genau dann einen Hamiltonkreis, wenn $L(\Pi_H) \neq \emptyset$.

Wenn wir bei einer Berechnung die Anzahl der Transitionen als Zeitmaß verstehen, ergibt sich für Π_H eine Worst-Case-Zeitkomplexität von $\mathcal{O}(n^3)$:

1. Alle Pfade der Länge n (und somit alle Hamiltonpfade) liegen nach $\lceil log_2\, n \rceil$ Transitionen in Region n' vor.

2. Ein solches Wort XwY ($w \in \underline{T}^*$, $|w| = n$) muss in jeder Region i' ($i \in [n-1]^+$) im ungünstigsten Fall $(n-1)$-mal rotiert werden. In Region n' liegen alle Pfade der Länge n vor, somit findet sich dort bereits zu Beginn die Form, bei der nur eine Rotation nötig ist. (Man beachte, dass Regel $r_{n',3}$ nur auf Wörter angewendet wird, bei denen ein Rotationsschritt vorgenommen wurde. Daher ist mindestens ein Rotationsschritt notwendig.) Insgesamt müssen maximal $1+(n-1)^2 = n^2-2n+2$ Rotationen ausgeführt werden. Wir runden diesen Wert auf Vielfache von n auf – die Begründung hierfür geben wir im nächsten Punkt – und erhalten somit n^2-n Rotationen (da $n \geq 2$).

3. Jede Rotation beginnt mit zwei Transitionen, in denen das rechte Symbol entfernt und links wieder angefügt wird. Da ein Hamiltonkreis jedes Symbol $i \in [n]^+$ genau einmal enthält, muss danach durchschnittlich $\frac{n-1}{2}$-mal der Index der Randsymbole dekrementiert werden, was jeweils zwei Regelanwendungen erfordert. Um diesen Durchschnitt zu garantieren, gehen wir bei der Gesamtzahl der erfolgten Rotationen von Vielfachen von n aus. Danach werden in zwei weiteren Transitionen die Randsymbole zu X und Y umgewandelt und im Zuge dessen bei Pfaden mit $\underline{i}$ an letzter Stelle auch die Ersetzung durch i vorgenommen. Pro Rotation rechnen wir demnach mit $n+3$ Transitionen.

4. Zum Schluss wird der Pfad noch im Verlaufe von zwei Regelanwendungen aus der Region 1_O an die Umgebung abgegeben.

Somit ergibt sich $n^3 + 2n^2 - 3n + \lceil log_2\, n \rceil + 2$ als obere Schranke für die Anzahl der benötigten Transitionen, um sämtliche Hamiltonkreise von G_H auszugeben. Man beachte zudem, dass es für Π_H keine Rolle spielt, über welchen mittleren Grad – d. h. welche durchschnittliche Anzahl von Kanten pro Knoten – G_H verfügt, obwohl dies exponentiellen Einfluss auf die Anzahl der Pfade einer bestimmten Länge in G_H haben kann.

7 Zusammenfassung und Bewertung

Wir haben in dieser Arbeit Splicing-P-Systeme vorgestellt, die eine Kombination aus Splicing-Systemen und P-Systemen darstellen. Letztere folgen einem *in-vivo*-Ansatz, indem sie eine von lebenden Zellen inspirierte Struktur und Funktionsweise aufweisen. Insofern sind Betrachtungen zu deren Berechnungsstärke zunächst nur theoretischer Natur, da keine Verfahren existieren, diese direkt in „Bioware“ umzusetzen. Eine Implementierung

mittels konventioneller Rechentechnik führt hingegen zu einer sequentiellen Simulation der maximal parallelen Arbeitsweise, weshalb ihre praktische Relevanz zu einem großen Teil in der Modellierung komplexer, meist biologischer Prozesse liegt.

Dieses Modell wurde daraufhin mit H-Systemen zusammengeführt, da uns mit dem Splicing eine Technik zur Verfügung steht, die eine massiv parallele Berechnung auf DNA *in vitro* ermöglicht. Wir haben mit dem für das Hamiltonkreisproblem gegebene System ein Beispiel dafür gegeben, wie Splicing-P-Systeme aufgrund der maximal parallelen Funktionsweise NP-vollständige Probleme in polynomieller Zeit lösen können.

Gegenüber reinen H-Systemen genügen zur Erlangung universeller Berechnungsstärke endliche Mengen von Splicingregeln und Axiomen. [PRS98] stellt ebenfalls einige Erweiterungen von H-Systemen vor, die mit endlichen Ressourcen über Turing-Mächtigkeit verfügen. Im Hinblick auf eine praktische Umsetzung ist dabei stets von Interesse, inwieweit die jeweiligen Modifikationen bei einer Umsetzung im Labor realisiert werden können. Für den Fall der Splicing-P-Systeme betrifft dies den durch die Zielindikatoren beschriebenen Transport der Axiome in unterschiedliche Regionen. Hierzu sind weitere Untersuchungen nötig, um zu entscheiden, ob Splicing-P-Systeme als mögliche Architektur eines DNA-Rechners in Frage kommen. Dann muss das im theoretischen Modell formulierte Vorhandensein der Axiome in unbegrenzter Anzahl natürlich zu einem Vorliegen „in ausreichender Anzahl“ abgeschwächt werden. Bis zu welcher Größenordnung ein solcher Rechner daher mit akzeptabler Zuverlässigkeit liefe, bliebe dabei ein stets abzuschätzendes Problem.

Aus mathematischer Sicht wurde mit Theorem 5.2 gezeigt, welche Klasse von Splicing-P-Systemen die Erzeugung aller rekursiv aufzählbaren Sprachen zulässt. Dabei bleibt offen, inwieweit die dort angegebenen Parameter minimal sind, ob also eine Klasse von Systemen mit geringerem Grad oder geringerem Durchmesser über die gleiche Ausdrucksstärke verfügt. Demnach ist auch nicht ausgeschlossen, dass Theorem 9.2 aus [Fri08] doch gezeigt werden kann. Wir haben bei der Korrektur auf eine dritte Membran zurückgegriffen und im Gegenzug den Durchmesser verringern können, allerdings bleibt eine Untersuchung vergleichbarer Resultate für den Grad 2 weiterhin interessant.

8 Literaturverzeichnis

[BGK+06] Francesco Bernardini, Marian Gheorghe, Natalio Krasnogor, Ravie C. Muniyandi, Mario J. Pérez-Jímenez, and Francisco José Romero-Campero. *On P Systems as a Modelling Tool for Biological Systems.* R. Freund et al. (Eds.): WMC 2005, LNCS 3850, pp. 114-133, 2006. Springer, 2006

[CPP06] Gabriel Ciobanu, Mario J. Pérez-Jímenez, and Gheorghe Păun. *Applications of Membrane Computing.* Springer Verlag 2006.

[Fri08] Pierluigi Frisco. *Computing with Cells.* Oxford Verlag, 2008.

[Pău02] Gheorghe Păun. *Membrane Computing: An Introduction.* Natural Computing Series. Springer, 2002.

[PR02] Gheorghe Păun and Grzegorz Rozenberg. "A guide to membrane computing." In: *Theoretical Computer Science* 287.1 (2002), pp. 73–100.

[PRS98] Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. *DNA Computing: New Computing Paradigms.* Springer, 1998

[PRS10] Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2010.

[PY99] Gheorghe Păun and Takashi Yokomori. "Membrane computing based on splicing." In: *E. Winfree and D. K. Gifford, editors, DNA Based Computers V, volume 54 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 217-232. American Mathematical Society, 1999.

[SCH01] Uwe Schöning. *Theoretische Informatik - kurzgefasst.* 4. Auflage. Spektrum, Akademischer Verlag, 2001

Eidesstattliche Erklärung

Hiermit erkläre ich, die am heutigen Tag eingereichte Arbeit „Splicing-P-Systeme“ selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.

Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, den 11. August 2016 .....

Benjamin Range