TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut Theoretische Informatik, Lehrstuhl für Automatentheorie

Bachelorarbeit

SPIKING NEURAL P-SYSTEME

Andy Püschel
Mat.-Nr.: 3760510

Gutachter:
Prof. Dr.-Ing. Franz Baader
Zweitgutachter:
Dr.-Ing. Monika Sturm
Eingereicht am 17. September 2015

ERKLÄRUNG

Ich erkläre, dass ich die vorliegende Arbeit selbständig, unter Angabe aller Zitate und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Dresden, 17. September 2015

INHALTSVERZEICHNIS

1 Einleitung **7**

2 Mathematische und sprachtheoretische Grundlagen **9**

3 SN P-Systeme **15**

3.1 Definition eines SN P-Systems 15

3.2 Aufbau eines SN P-Systems 17

3.3 Funktionsablauf 18

3.4 Interpretation der Ein- und Ausgänge 19

3.4.1 Generativer Modus 19

3.4.2 Akzeptierender Modus 20

3.4.3 Funktionaler Modus 21

3.5 Beispiel 21

4 Modifikationen von SN P-Systemen **25**

4.1 Modifikation bezüglich der Regeln 26

4.1.1 SN P-Systeme mit restriktiven Regeln 27

4.1.2 Spiking Neural P-Systeme mit erweiterten Regeln 28

4.2 Modifikation bezüglich der Form 30

4.2.1 Homogene Spiking Neural P-Systeme 30

4.2.2 Zeitunabhängige Spiking Neural P-Systeme 32

4.3 Modifikation bezüglich des Verhaltens 33

4.3.1 Synchrone Spiking Neural P-Systeme 33

4.3.2 Asynchrone Spiking Neural P-Systeme 34

4.3.3 Sequenzielle Spiking Neural P-Systeme 35

5 Zusammenfassung **37**

6 Literaturverzeichnis **39**

1 EINLEITUNG

Natural Computing weist auf die Berechnungsprozesse in der Natur und auf die von der Natur inspirierten und vom Mensch geschaffenen Berechnungsprozessen hin. Bereits in [PŌ0] wurde *Membrane Computing* als eine Ausprägung des *Natural Computing* untersucht. *Membrane Computing* beginnt bei der Betrachtung von kleinen biologischen Apparaturen, den Zellen. In der Struktur und in der Funktionsweise einer biologischen Zelle spielt die Membran eine entscheidende Rolle. So trennt die Membran nicht nur die Zellen untereinander, sondern dient auch als eine Transportschicht zwischen den Zellen, um Informationen in der Form von chemischen Stoffen auszutauschen. Mit P-Systemen wurde erstmals eine Abstraktion von biologischen Zellen als eine mögliche Modellierung geliefert. Dabei wurden Komponenten innerhalb einer Zelle berücksichtigt, die durch eine Membran separiert wurden. Als Beispiel dient hierzu der Zellkern, der von einer Zellmembran vom umliegenden Zellplasma innerhalb einer biologischen Zelle abgeschirmt ist. In [PŌ2] wurde in diesem Zusammenhang die biologische Nervenzelle betrachtet. Die biologische Nervenzelle, auch Neuron genannt, besitzt neben dem Zellkörper (Soma), welcher von einer Membran umhüllt ist, noch weitere Komponenten. Neuronen sind für die Aufgabe ausgelegt, Informationen von einer Zelle zu weiteren Zellen zu schicken. Dazu besitzen Neuronen zusätzlich Dendriten, die sich direkt am Zellkörper befinden, Synapsen und ein Axon, das eine Verbindung zwischen dem Zellkörper und den Synapsen darstellt. Ein Neuron empfängt Informationen in Form von elektrischen Impulsen von den Dendriten. Nach Verarbeitung dieser Informationen sendet das Neuron über das Axon einen elektrischen Impuls zu den Synapsen. Ob ein elektrischer Impuls gesendet wird, hängt vom Aktionspotenzial des Neurons ab. Das Aktionspotenzial ist eine Schwelle, die mittels empfangener elektrischer Impulse überschritten werden muss, damit das Neuron in Aktion tritt und ein Signal sendet. Dieses elektrische Signal wird von den Synapsen aus zu weiteren Neuronen gesendet. Ein Versuch, aus einem konventionellen P-System nun eine Abstraktion für ein neuronales Netz herzuleiten, wurde bereits in [IPY06] angegeben. Diese Abstraktion von neuronalen Netzen des biologischen Nervensystems wird als *Spiking Neural* P-System (oder kurz als SN P-System) bezeichnet. Hierbei steht nun nicht mehr die Kommunikation der durch die Membran getrennten Komponenten innerhalb einer Zelle im Vordergrund, sondern die Kommunikation verschiedener Zellen untereinander.
Ziel dieser Arbeit ist es, ein Allgemeinverständnis von SN P-Systemen im Hinblick auf die Definition ihrer formalen Struktur, die Funktionsweise und den Aufbau von SN P-Systemen zu vermitteln und dabei auf die unterschiedlichen Modifikationen der SN P-Systeme unter der Betrachtung ihrer Berechnungsstärke einzugehen. Beispiele sollen in den jeweiligen Kapiteln zum Verständnis der einzelnen modifizierten SN P-Systeme beitragen. Dafür werden im Kapitel 2 zunächst die mathematischen und sprachtheoretischen Grundlagen geklärt, um in Kapitel 3 ein SN P-System zu definieren, sowie den Aufbau und die Funktionsweise zu beschreiben. In Kapitel 4 werden verschiedene Modifikationen von SN P-Systemen, die der Beschreibung eines realen Nervensystems nahe kommen, vorgestellt und hinsichtlich ihrer Berechnungsstärke untersucht. Noch offene Fragen oder weitere Modifikationen werden in Kapitel 5 vorgestellt.

2 MATHEMATISCHE UND SPRACHTHEORETISCHE GRUNDLAGEN

Zur Unterscheidung zwischen der Menge der natürlichen Zahlen ohne der Zahl 0 und der Menge der natürlichen Zahlen inklusive der Zahl 0 werden für die vorliegende Arbeit die folgenden Notationen festgelegt:

Notation 2.0.1. Die Menge der nichtnegativen ganzen Zahlen inklusive 0 wird mit $\mathbb{N}_0$ notiert[1], wobei $\mathbb{N}_0 := \{0, 1, 2, ...\}$.

Notation 2.0.2. Die Menge der nichtnegativen ganzen Zahlen größer 0 wird mit $\mathbb{N}_1$ notiert, wobei $\mathbb{N}_1 := \mathbb{N}_0 \backslash \{0\}$.

Um ein SN P-System vollständig beschreiben und die in dieser Arbeit verwendeten Beispiele erläutern zu können, bedarf es der Verwendung folgender sprachtheoretischer Begriffe nach [Baa13] und [Baa14]:

Definition 2.0.1 (Potenzmenge). Sei A eine beliebige Menge, dann ist 2^A die Potenzmenge von A, die A und alle Teilmengen von A enthält. Dabei gilt:

$$2^A := \{B \mid B \subseteq A\} \cup \{\emptyset\}$$

Definition 2.0.2 (Alphabet). Ein *Alphabet* Σ ist eine nichtleere Menge von Symbolen. Die Menge aller Folgen von Symbolen von Σ wird als Σ^* bezeichnet.

Definition 2.0.3 (Wort). Ein *Wort* w über einem Alphabet Σ ist eine endliche Folge von Symbolen $w = a_1...a_n$ mit $a_i \in \Sigma$. Dabei wird das *leere Wort* mit ε notiert.

Definition 2.0.4 (Wortlänge). Sei $w = a_1...a_n$ ein Wort über dem Alphabet Σ, so wird die *Länge des Wortes* w als $|w|$ bezeichnet, wobei $|w| = n$ und $|w| = 0$ für $w = \varepsilon$.

Definition 2.0.5 (regulärer Ausdruck). Sei Σ ein endliches Alphabet. Dann ist die Menge Reg_Σ der *regulären Ausdrücke* über Σ induktiv definiert:

- $\emptyset$ ist ein regulärer Ausdruck, der die leere Menge $\emptyset$ beschreibt.
- ε ist ein regulärer Ausdruck, der die Menge $\{\varepsilon\}$ beschreibt.

[1] Notation nach [Pea94]

- $\forall x \in \Sigma$ ist x ein regulärer Ausdruck, der die Menge $\{x\}$ beschreibt.
- Wenn r und s reguläre Ausdrücke sind, so sind $(r|s)$ (Alternative) und $(r \cdot s)$ (Konkatenation) reguläre Ausdrücke.
- Ist r ein regulärer Ausdruck, so ist r^* (Iteration) ebenfalls ein regulärer Ausdruck.

Notation 2.0.3. Zur Vereinfachung werden für zwei reguläre Ausdrücke r und s notiert:

$$\begin{aligned} (r + s) &:= (r|s) \\ (rs) &:= (r \cdot s) \\ r^0 &:= \varepsilon \\ r^+ &:= r \cdot r^* \\ r^n &:= \underbrace{r \cdot \ldots \cdot r}_{n \text{ mal}} \qquad | \text{ mit } n \in \mathbb{N}_1 \end{aligned}$$

Entsprechend der Prioritäten der Operationssymbole können überflüssige Klammern entfallen. Dabei haben *, $^+$, ^{n} höchste Priorität, gefolgt von $\cdot$ und der niedrigsten Priorität von $+$.

Definition 2.0.6 (NEA). Ein *nichtdeterministischer endlicher Automat* (nachfolgend mit NEA abgekürzt) $\mathcal{A}$ ist ein 5-Tupel,

$$\mathcal{A} := (Q, \Sigma, q_0, \delta, F),$$

wobei gilt:

- Q ist die endliche Menge der Zustände.
- Σ ist das endliche Eingabealphabet.
- q_0 ist der Startzustand (Initialzustand).
- $\delta : Q \times \Sigma \to 2^Q$ ist die Transitionsfunktion.
- F ist die Menge der Finalzustände.

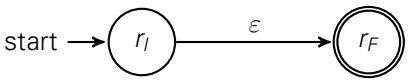
Definition 2.0.7 (Pfad). Ein *Pfad* in einem NEA $\mathcal{A}$ ist eine Folge $\pi = (q_0, a_1, q_1)(q_1, a_2, q_2)\ldots(q_{n-1}, a_n, q_n)$ mit $(q_i, a_{i+1}, q_{i+1}) \in \delta$ für $i = 0, \ldots, n-1$, $a_i \in \Sigma$ und $q_i \in Q$. π ist ein Pfad von q_0 nach q_n.
Die Beschriftung des Pfades π ist das Wort $w := a_1 \ldots a_n$.
Die Länge des Pfades π ist n.
Für $n = 0$ spricht man vom leerem Pfad, welcher die Beschriftung ε hat.
Die Existenz eines Pfades in $\mathcal{A}$ von p nach q mit Beschriftung w wird durch $p \xrightarrow{w}_{\mathcal{A}} q$ beschrieben.

Zur besseren Veranschaulichung wird für die Definition 2.0.8 die folgende Notation eingeführt: Sei r ein beliebiger regulärer Ausdruck und $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein NEA, so dass $L(\mathcal{A}) = L(r)$ gilt. So gelte die folgende Notation: $q_0 = r_I$ und $q_n = r_F$ mit $q_n \in F$. Des Weiteren werden zur besseren Veranschaulichung noch zu konstruierende NEA's als ein Start- und ein Finalzustand, die mit einem gestrichelten Pfeil verbunden sind, dargestellt.

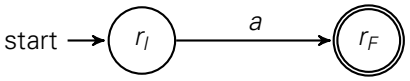
Definition 2.0.8 (Thompson-Konstruktion[2] nach [HU79]). Für einen regulären Ausdruck r kann effizient ein NEA $\mathcal{A}$ mit ε-Übergängen mittels der *Thompson-Konstruktion* entwickelt werden, so dass $L(\mathcal{A}) = L(r)$ gilt. Sei $a \in \Sigma$ und $s, t \in Reg_\Sigma$ reguläre Ausdrücke über dem Alphabet Σ, so ist die Konstruktionsvorschrift induktiv definiert:

$r = \emptyset$

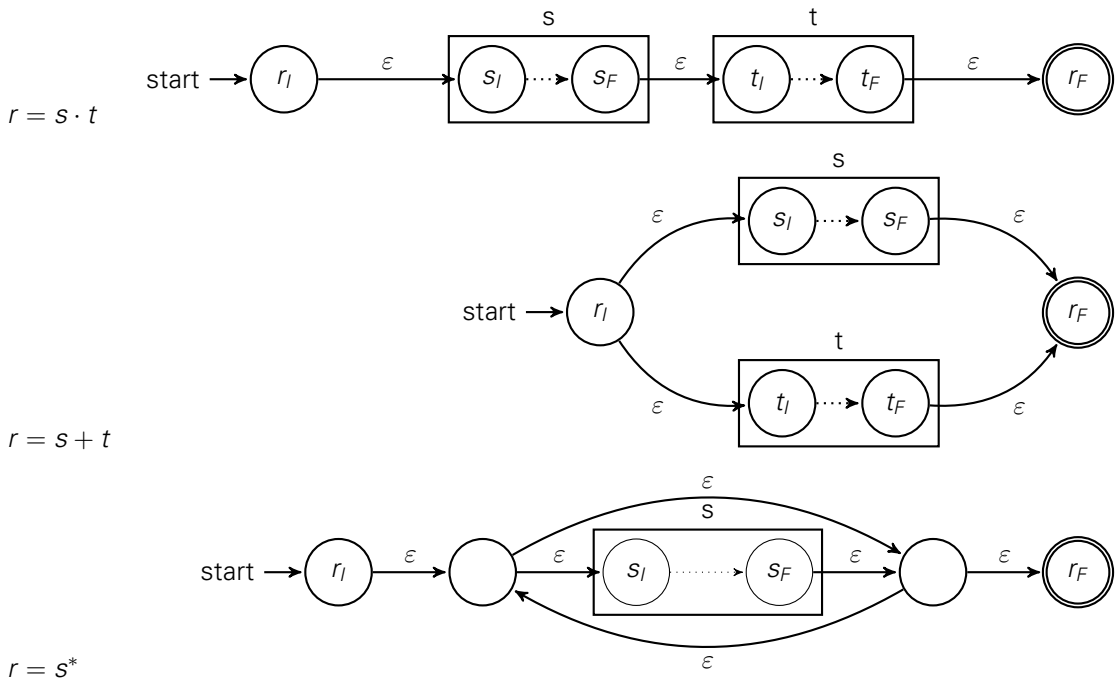
$r = \varepsilon$



$r = a$



[2] Grundidee der Konstruktion aus [Tho68]



Der entstandene NEA $\mathcal{A}'$ mit ε -Übergängen ist von der Form $\mathcal{A}' = (Q, \Sigma, q_0, \delta, F')$. Der NEA $\mathcal{A}$ mit ε -Übergängen ist eine Modifikation von $\mathcal{A}'$, wobei gilt:

$$\mathcal{A} := (Q, \Sigma, q_0, \delta, F)$$
$$\text{mit} \qquad F := \{q \in F' \mid \nexists (q, a, p) \in \delta \text{ mit } p \in Q, a \in \Sigma \cup \{\varepsilon\}\}$$

Definition 2.0.9 (ε-Eliminierung)**.** Sei $\mathcal{A}' = (Q, \Sigma, q_0, \delta', F')$ ein NEA mit ε- Übergängen. So ist die *ε-Eliminierung* eine Konstruktionsvorschrift, die zu $\mathcal{A}'$ einen äquivalenten NEA $\mathcal{A}$ liefert, so dass $L(\mathcal{A}) = L(\mathcal{A}')$ gilt. Der NEA $\mathcal{A}$ ohne ε-Übergänge wird dabei wie folgt konstruiert:

$$\mathcal{A} := (Q, \Sigma, q_0, \delta, F)$$
$$\delta := \{(p, a, q) \in Q \times \Sigma \times Q \mid p \xrightarrow{a}_{\mathcal{A}} q\}$$
$$F' := \begin{cases} F \cup \{q_0\} & \text{falls } q_0 \xrightarrow{\varepsilon}_{\mathcal{A}} q \\ F & \text{sonst} \end{cases}$$

Definition 2.0.10 (formale Sprache)**.** Eine *formale Sprache* L über dem Alphabet Σ ist eine Menge von Wörtern mit $L \subseteq \Sigma^*$.
Für die formalen Sprachen L, L_1 und L_2 sind folgende Operationen definiert:

Konkatenation: $L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$

Quotient: $L_1/L_2 := \{w \in \Sigma^* \mid \exists z \in L_2 \text{ mit } wz \in L_1\}$

Kleensche Hülle:
$$L^0 := \{\varepsilon\}$$
$$L^n := \bigcup_{i \geq 0}^{n} L^i$$
$$L^* := \bigcup_{i \geq 0} L^i$$
$$L^+ := L^* \backslash L^0$$

Komplement: $L^C := \Sigma^* \backslash L$

Definition 2.0.11. Seien r und s reguläre Ausdrücke und $a \in \Sigma$ ein Symbol, so ist die formale Sprache L wie folgt definiert:

$$L(\emptyset) := \emptyset \qquad L(\varepsilon) := \{\varepsilon\}$$
$$L(a) := \{a\} \qquad L(r^*) := L(r)^*$$
$$L(r + s) := L(r) \cup L(s) \qquad L(r \cdot s) := L(r) \cdot L(s)$$

Definition 2.0.12 (FIN). Eine Sprache L ist Element der Klasse der endlichen Sprachen *FIN* mit $L \in FIN$ und heißt endlich, wenn die Sprache L nur endlich viele Wörter w_i enthält, mit $L = \{w_1, \ldots, w_n\}$.

Definition 2.0.13 (TM). Eine *Turingmaschine* über dem Eingabealphabet Σ ist von der Form $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \Delta, F)$, wobei

- Q die endliche Zustandsmenge ist,
- Σ das Eingabealphabet ist,
- Γ das Arbeitsalphabet ist mit $\Sigma \subseteq \Gamma$, $\not b \in \Gamma \backslash \Sigma$,
- $q_0 \in Q$ der Anfangszustand ist,
- $F \subseteq Q$ die Finalzustandsmenge ist und
- $\Delta \subseteq Q \times \Gamma \times \Gamma \times \{r, l, n\} \times Q$ die Übergangsrelation ist.

Dabei bedeutet $(q, a, a', d, q') \in \Delta$ mit $d \in \{r, l, n\}$:
Im Zustand q mit a auf dem gerade gelesenen Feld (Arbeitsfeld) kann die Turingmaschine $\mathcal{A}$ das Symbol a durch a' ersetzen, in den Zustand q' gehen und den Schreib-Lesekopf entweder um ein Feld nach rechts ($d = r$), links ($d = l$) oder nicht ($d = n$) bewegen.
Die Maschine $\mathcal{A}$ heißt deterministisch, falls es für jedes Tupel $(q, a) \in Q \times \Gamma$ höchstens ein Tupel der Form $(q, a, a', d, q') \in \Delta$ gibt.
NTM steht im Folgenden für (möglicherweise nichtdeterministische) Turingmaschinen und DTM für deterministische Turingmaschinen.

Definition 2.0.14 (Turing-berechenbar). Eine (partielle) Funktion $f : (\Sigma^*)^n \to \Sigma^*$ (bzw. $\mathbb{N}_0^n \to \mathbb{N}_0$ in einer geeigneten Kodierung) heißt Turing-berechenbar, wenn es eine DTM $\mathcal{A}$ gibt, mit

- $\forall x_1, \ldots, x_n \in \Sigma^* : \not b q_0 x_1 \not b \ldots \not b x_n \not b \vdash_{\mathcal{A}}^* k$ mit k Stoppkonfiguration gdw. $(x_1, \ldots, x_n) \in dom(f)$ (Definitionsbereich von f).
- Im Fall $(x_1, \ldots, x_n) \in dom(f)$ muss die Beschriftung des Bandes in der Stoppkonfiguration k rechts vom Schreib-Lesekopf bis zum ersten Symbol $\notin \Sigma$ der Wert $y = f(x_1, \ldots, x_n)$ der Funktion sein, d.h. k muss die Form $uqyv$ haben mit
  - $v \in (\Gamma \backslash \Sigma) \cdot \Gamma^* \cup \{\varepsilon\}$
  - $y = f(x_1, \ldots, x_n)$.

Definition 2.0.15 (Turing-akzeptierbar). Eine Sprache L heißt Turing-akzeptierbar (rekursiv aufzählbar), wenn es eine NTM $\mathcal{A}$ gibt mit $L = L(\mathcal{A})$.

Definition 2.0.16 (rekursiv aufzählbare Menge). Eine Menge natürlicher Zahlen M heißt rekursiv aufzählbar, wenn $M = \emptyset$ oder es eine Turing-berechenbare Funktion $f : \mathbb{N}_0^n \to \mathbb{N}_0$ gibt, so dass $M = ran(f)$ (Wertebereich von f) gilt.

Definition 2.0.17 (NRE). Sei M eine rekursiv aufzählbare Menge natürlicher Zahlen, so ist *NRE* die Klasse der rekursiv aufzählbaren Mengen natürlicher Zahlen, mit $M \in NRE$.

Definition 2.0.18 (RE). Sei L eine Turing-akzeptierbare (rekursiv aufzählbare) Sprache, so ist *RE* die Klasse der Turing-akzeptierbaren Sprachen, mit $L \in RE$.

Neben den Turing-Maschinen sind auch Registermaschinen in der Lage, Turing-berechenbare Funktionen zu berechnen und Turing-akzeptierbare Sprachen zu akzeptieren.

Definition 2.0.19 (Registermaschine aus [IPY06][3]). Eine *Registermaschine* ist von der Form $\mathcal{M} = (m, H, l_0, l_H, I)$, wobei

[3] Mihai Ionescu verweist hierbei auf [Min67]

- m die Anzahl der Register ist,
- H die endliche Menge der Markierungen von ausführbaren Anweisungen ist,
- l_0 die Anfangsmarkierung ist,
- l_H die Haltemarkierung ist und
- I die Menge der ausführbaren Anweisungen (Instruktionen) ist.

Jeder Markierung aus H wird genau eine Instruktion aus I zugeordnet. Die Instruktionen folgen den Markierungen, separiert durch :, und sind von der folgenden Form:

- $l_i : (ADD(r), l_j, l_k)$ addiert den Wert 1 zu dem im Register $r \in \mathbb{N}_0$ mit $r < m$ befindlichen Wert hinzu, und geht anschließend nichtdeterministisch zu einer der Instruktionen mit der Markierung $l_j \in H$ oder $l_k \in H$.
- $l_i : (SUB(r), l_j, l_k)$ subtrahiert den Wert 1 vom Wert v im Register $r \in \mathbb{N}_0$ mit $r < m$, wenn $v > 0$ gilt und geht anschließend zur Instruktion mit der Markierung $l_j \in H$, oder geht zur Instruktion mit der Markierung $l_k \in H$, wenn $n = 0$ gilt.
- $l_h : HALT$ hält die Registermaschine mit dem Ergebnis im Register 0.

Eine Registermaschine $\mathcal{M}$ berechnet eine Zahl n auf folgende Weise: Jedes Register beinhaltet zum Start der Ausführung der Registermaschine $\mathcal{M}$ den Wert 0 (die Register werden leer genannt). Im nächsten Schritt wird die Instruktion mit der Markierung l_0 angewendet. Wird eine Instruktion der Form $l_i : (ADD(r), l_j, l_k)$ oder $l_i : (SUB(r), l_j, l_k)$ angewendet, so wird im darauffolgenden Schritt die Instruktion mit der Markierung l_j oder l_k angewendet. Wird die Instruktion mit der Markierung l_H erreicht, so hält die Registermaschine mit dem Wert n im Register 0 als Ergebnis von $\mathcal{M}$.
Die Registermaschine heißt deterministisch, wenn für alle Vorkommen von $l_i : (ADD(r), l_j, l_k)$ gilt: $l_j = l_k$, sonst heißt sie nichtdeterministisch.

3 SN P-SYSTEME

Spiking Neural P-Systeme (mit SN P-Systeme abgekürzt) stellen eine Abstraktion von einem Nervensystem dar, für die der Informationsaustausch zwischen den einzelnen Neuronen formal beschrieben wird. Dabei werden hauptsächlich das Neuron als Hauptkörper, die ausgehenden Synapsen des Axons und das Aktionspotenzial des Neurons als eine elektrische Erregung betrachtet.

3.1 DEFINITION EINES SN P-SYSTEMS

Der elektrische Impuls des Aktionspotenzials bildet das einzige Element a als Repräsentant des *Alphabets* $O = \{a\}$. Folgend wird eine modifizierte Form des SN P-Systems aus [IPY06] als ein unbestimmtes SN P-System definiert:

Definition 3.1.1 (SN P-System). Ein *SN P-System* Π ist ein Tupel der Form:

$$\Pi = (O, \sigma_1, ..., \sigma_m, syn, in, out),$$

wobei gilt:

- O ist das einelementige *Alphabet* mit $O = \{a\}$, wobei a den elektrischen Impuls (*Spike*) simuliert.
- $\sigma_1, ..., \sigma_m$ sind die Neuronen der Form

  $$\sigma_i = (n_i, R_i) \text{ mit } i \in \mathbb{N}_1,\ 1 \leq i \leq m,$$

  mit:
  - n_i ist die Startanzahl an *Spikes* im Neuron i.
  - R_i ist eine Menge von Regeln im Neuron i von der Form $(E/v \rightarrow w; d)$ oder $(u \rightarrow \lambda; d)$,
    wobei E ein regulärer Ausdruck über dem Alphabet O, $d \in \mathbb{N}_0$ und $u, v, w \in O^*$ ist mit den Bedingungen:
    1. $u \notin L(E)$ im Neuron i, damit nie eine Regel der Form $(E/v \rightarrow w; d)$ angewendet werden kann, wenn es möglich ist, eine Regel der Form $(u \rightarrow \lambda; d)$ anzuwenden.
    2. Wenn für die Sprachen zweier regulärer Ausrücke E_1 und E_2 nun $L(E_1) \cap L(E_2) \neq \emptyset$ gilt, so kann nichtdeterministisch für eine Anzahl von Spikes n die Regel $(E_1/v \rightarrow w; d)$ oder die Regel $(E_2/v \rightarrow w; d)$ mit $a^n \in L(E_1) \cap L(E_2)$ und $a \in \Sigma$ gewählt werden.

- *syn* ist die Menge der gerichteten Transitionen (*Synapsen*) zwischen den Neuronen von der Form $syn \subseteq \{1, ..., m\} \times \{1, ..., m\}$ mit $(i, i) \notin syn$, $i \in \mathbb{N}_1$ und $i \leq m$.
- *in* ist der Index des Input-Neurons mit $in \in \mathbb{N}_0$ und $in \leq m$. Ist kein Input-Neuron vorhanden, so wird für *in* der Wert 0 notiert.
- *out* ist der Index des Output-Neurons mit $out \in \mathbb{N}_0$ und $out \leq m$. Ist kein Output-Neuron vorhanden, so wird für *out* der Wert 0 notiert.

Um die in dieser Arbeit verwendeten Beispiele und Vertiefungen zu veranschaulichen, werden des Weiteren für das SN P-System und seine Umgebung (Kontext) folgende Begriffe geklärt:

Definition 3.1.2 (Zeit). t ist die *Zeit des SN P-Systems* beginnend bei 0. Sie erhöht sich beim Erreichen einer Folgekonfiguration um 1.

Definition 3.1.3 (Zustand). Sei σ_i ein Neuron, so ist s_i der Zustand des Neurons σ_i. s_i ist dabei ein Tupel:

$$s_i := (c_i, \tau_i(t))$$

wobei gilt:

- c_i ist die aktuelle Anzahl der *Spikes* im Neuron σ_i, mit $c_i \in \mathbb{N}_0$.
- $\tau_i(t) : \mathbb{N}_0 \mapsto \mathbb{N}_0$ berechnet die verbleibende Zeit - in Abhängigkeit der zuletzt zum Zeitpunkt t_0 angewendeten Regel $r_i = E/v \rightarrow w; d$ - bis das Neuron σ_i wieder Spikes empfangen und Regeln anwenden kann, mit

$$\tau_i(t) := \begin{cases} 0 & \text{, wenn } t \geq t_0 + d \text{ oder } d \text{ ist undefiniert} \\ t - t_0 + d & \text{, sonst.} \end{cases}$$

Man nennt ein Neuron σ_i geschlossen, wenn es sich im Zustand $s_i = (c_i, \tau_i(t))$ mit $\tau_i(t) > 0$ befindet. Für den Fall $\tau_i(t) = 0$ nennt man das Neuron σ_i geöffnet.

Nach [IPY06] setzt sich die Konfiguration des SN P-Systems aus zwei Komponenten zusammen.

- Zum einen die Verteilung der *Spikes* auf die Neuronen und
- zum anderen der Zustand der Neuronen in Abhängigkeit der Verzögerung der zuletzt verwendeten Regel in jedem Neuron.

Definition 3.1.4 (Konfiguration). Sei Π ein SN P-System mit $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$, so ist eine Konfiguration C die Verteilung der *Spikes* in den Neuronen in Abhängigkeit des Zustands der Neuronen:

$$C := \langle s_1, \ldots, s_m \rangle$$

wobei gilt:

- s_i ist der derzeitige Zustand des Neurons σ_i mit $1 \leq i \leq m$.

Die Anfangskonfiguration wird mit C_0 notiert, wobei $C_0 = \langle s_1, \ldots, s_m \rangle$ mit $\sigma_i = (n_i, R_i)$ und $1 \leq i \leq m$, $s_i = (n_i, 0)$.
Gelangt man von einer Konfiguration C_i direkt zu einer Folgekonfiguration C_{i+1}, so wird der Übergang von C_i nach C_{i+1} als Transition bezeichnet und mit $C_i \Rightarrow C_{i+1}$ notiert.

Definition 3.1.5 (Berechnung nach [PPJR06]). Sei $C_i \Rightarrow C_{i+1}$ eine Transition, so ist eine Sequenz von Transitionen, beginnend mit der Startkonfiguration, eine Berechnung γ, mit:

$$\gamma := C_0 \Rightarrow C_1 \Rightarrow C_2 \Rightarrow \dots$$

Dabei ist der i-te Schritt von γ eine Transition $C_{i-1} \Rightarrow C_i$ in γ, wobei $i \in \mathbb{N}_1$. Die Menge aller Berechnungen von einem SN P-System Π wird mit $COM(\Pi)$ notiert. Die Menge aller haltenden Berechnungen von Π wird mit $HCOM(\Pi)$ notiert. Dabei gilt: $\gamma \in COM(\Pi)$ und $HCOM(\Pi) \subseteq COM(\Pi)$.

Definition 3.1.6 (Spiketrain). Sei $\gamma = C_0 \Rightarrow C_1 \Rightarrow C_2 \Rightarrow \dots$ eine Berechnung von einem SN P-System Π, so ist $st(\gamma)$ der zu γ gehörende *Spiketrain*, welcher Informationen t_i zu den in die Umgebung gesendeten Spikes der einzelnen Konfigurationen C_i enthält:

$$st(\gamma) := \langle t_1, t_2, \dots \rangle \text{ mit } t_i \in \mathbb{N}_0$$

Erreicht γ eine Haltekonfiguration oder werden nur endlich viele Spikes in die Umgebung gesendet, so ist $st(\gamma)$ endlich, ansonsten unendlich. Dabei ist $ST(\Pi) = \{st(\gamma) \mid \gamma \in COM(\Pi)\}$ die Menge aller Spiketrains über Π mit $st(\gamma) \in ST(\Pi)$.

Notation 3.1.1. Die Informationen in einem Spiketrain $st(\gamma)$ können entsprechend der Untersuchung nach bestimmten Kriterien im Hinblick auf spezifische Ergebnismengen betrachtet werden. In [PPJR06] wurde im Allgemeinem der Spiketrain im Zusammenhang auf natürliche Zahlen untersucht, wofür die Zeitpunkte, in welchen mindestens ein *Spike* in die Umgebung gesendet wurde, berücksichtigt wurden. Daher wird für die Information im *Spiketrain* der Bezeichner t_i verwendet. Um die nach spezifischen Kriterien entsprechenden Untersuchungen zu differenzieren, werden die folgenden Notationen für einen Spiketrain $st(\gamma) = \langle t_1, t_2, \dots \rangle$ in dieser Arbeit eingeführt:

- $st_N(\gamma)$ ist eine Sequenz der Zeitpunkte t_i, für die eine Konfiguration C_{t_i} von der Berechnung γ mindestens einen *Spike* in die Umgebung sendet (nach [PPJR06]).
- $st_S(\gamma)$ ist eine Sequenz der Anzahlen von Spikes t_i, die in den Konfigurationen C_i von der Berechnung γ aus einem Output-Neuron in die Umgebung gesendet werden.
- $st_B(\gamma)$ ist eine binäre Sequenz mit $t_i \in \{0, 1\}$. Es gilt: $t_i = 1$, wenn in C_i mindestens ein Spike in die Umgebung gesendet wird, sonst gilt $t_i = 0$.

3.2 AUFBAU EINES SN P-SYSTEMS

Die einzelnen Neuronen σ_i werden in dieser Arbeit durch Rechtecke mit abgerundeten Ecken dargestellt. Diese Neuronen sind mittels Transitionen, welche als gerichtete Pfeile eines Graphen dargestellt werden, gemäß der Definition des SN P-Systems Π untereinander vernetzt. Jedes Neuron σ_i besitzt eine Startanzahl von *Spikes* n_i und eine Menge von Regeln R_i mit $i \in \mathbb{N}_1$ und $i \leq m$, welche die Bedingungen zum Feuern eines Signals angeben.
Das Initialwort n_i der Form $n_i = a^k$ mit $k \in \mathbb{N}_0$ kann entfallen, wenn $k = 0$ gilt, ansonsten wird es in der ersten Zeile des Neurons angegeben, gefolgt von den Regeln r_j mit $j \in \mathbb{N}_1$ in den weiteren Zeilen. Eine Regel $r_j \in R_i$ eines Neurons σ_i ist entsprechend der beiden Möglichkeiten eine *Firing Rule* (Tabelle 3.2.1) oder eine *Forgetting Rule* (Tabelle 3.2.2). Gilt $L(E_j) = \{v_j\}$ für den regulären Ausdruck E_j aus der Regel r_j, so kann für die Regel r_j die Angabe von E_j entfallen.
Jedes Neuron σ_i wird mit seinem derzeitigen Zustand s_i markiert, wobei zum Start des SN P-Systems alle Neuronen geöffnet sind. Der Zustand des Gesamtsystems wird als Konfiguration C bezeichnet. Das System befindet sich zum Start der Simulation in der Startkonfiguration C_0. Wird eine Konfiguration erreicht, bei welcher es nicht möglich ist, eine weitere Regel in R_i aus einem beliebigen Neuron σ_i anzuwenden, so hält das System und eine Haltekonfiguration C_H ist erreicht.
Für das Gesamtsystem gilt die Zeit t, welche als globale Uhr fungiert. Zur Synchronisation des Gesamtsystems orientieren sich die Verzögerungen d_j aus $r_j \in R_i$ des Neurons σ_i an t.
Der Output erfolgt mittels einer Transition in die Umgebung vom Neuron σ_i nach dem Index *out* des Output-Neurons. Der Input erfolgt mit einer Transition von der Umgebung zu dem Neuron, welches mit dem Index *in* als Input-Neuron festgelegt wurde.

Tabelle 3.2.1: Firing Rule mit Neuron σ_i

$E/v \rightarrow w; d$	Form
E	ist der regulärer Ausdruck über dem Alphabet O.
v	ist das zu konsumierende Wort mit $\|v\| \leq n_i$.
w	ist das zu sendende Wort an alle benachbarten Folgeneuronen σ_j mit $(i, j) \in syn$.
d	ist die Verzögerung bis zum Senden des Impulses der gefeuerten Regel.

Tabelle 3.2.2: Forgetting Rule mit Neuron σ_i

$u \rightarrow \lambda; d$	Form
u	ist das zu konsumierende Wort. Anwendung kann nur erfolgen, wenn $n_i = \|u\|$.
λ	zeigt das Verwerfen des Wortes u ohne dem Senden eines Impuls.
d	ist die Verzögerung bis zum Verwerfen des Wortes der gefeuerten Regel.

3.3 FUNKTIONSABLAUF

Eine Regel $r_j \in R_i$ des Neurons σ_i mit $i, j \in \mathbb{N}_1$ kann in Abhängigkeit des Zustandes s_i angewendet (gefeuert) werden, wenn das Neuron σ_i ein *Wort* n_i besitzt, so dass $n_i \in L(E_j)$ mit $r_j = E_j/v \rightarrow w; d$ (*Firing Rule*) oder eine Regel der Form $r_j = u \rightarrow \lambda$ mit $n_i = u$ (*Forgetting Rule*) gilt. Dabei beschreiben v bzw. u das zu konsumierende Wort von n_i mit $\exists x \in O^*.n_i = |v| + |x|$ und w das zu sendende Wort w vom Neuron σ_i zu allen benachbarten Neuronen σ_j, für welche es eine Transition $(i, j) \in syn$ gibt, nach d Zeiteinheiten.
Ein Neuron σ_i kann ein Wort w empfangen und eine Regel $r_j \in R_i$ anwenden, wenn das Neuron geöffnet ist. Nach dem Anwenden der Regel r_j schließt sich das Neuron bis es nach d_j Zeiteinheiten den resultierenden Impuls aus r_j sendet und sich wieder öffnet.
Während ein Neuron geschlossen ist, kann es weder weitere Regeln anwenden, noch Wörter anderer Neuronen empfangen. Jedes Neuron σ_i, welches nicht geschlossen ist, empfängt das Wort w_j vom Neuron σ_j, sofern es eine Transition $(i, j) \in syn$ gibt. So können *Symbole* vervielfältigt werden. Dabei gilt die folgende Reihenfolge:

1. Falls ein Neuron σ_i geöffnet ist, wird, sofern möglich, eine Regel $r_j \in R_i$ zum Zeitpunkt t mit $t_{0_i} := t$ angewendet.
2. Falls eine Regel r_j angewandt wurde, wird mit v_j aus r_j die Anzahl der *Spikes* verringert:
$$n_i := n_i - |v_j|.$$
3. Die Anzahl der *Spikes* eines Neurons σ_i wird entsprechend der empfangenen Wörter w_j von benachbarten Neuronen erhöht:
$$n_i := n_i + \sum_{j=1}^{m} |w_j| \text{ mit } (j, i) \in syn.$$
4. Falls eine Regel aus R_i angewendet wurde, ist das Neuron σ_i nun geschlossen.
5. Das Neuron σ_i sendet das Wort w_i der angewandten Regel aus R_i zum Zeitpunkt $t = t_{0_i} + d_i$, wobei d_i die Verzögerung der angewendeten Regel aus R_i ist.
6. Das Neuron σ_i ist nun wieder geöffnet.

Für jeden Zeitpunkt t werden entlang der Transitionen zwischen den Neuronen ein Wort als Signal gesendet. Kann kein Wort aufgrund des Zustands s gesendet werden, so wird das Wort a^0 übermittelt.

3.4 INTERPRETATION DER EIN- UND AUSGÄNGE

Entsprechend der vier Möglichkeiten für *in* und *out*, unterscheidet man zwischen den verschiedenen Modi, die ein SN P System annehmen kann. Dabei wird die Möglichkeit für $in = 0$ und $out = 0$ vernachlässigt, da mit diesen SN P Systemen weder Berechnungen, noch Überprüfungen durchgeführt werden können.

3.4.1 Generativer Modus

Ein SN P Systen im generativen Modus empfängt keine *Spikes* aus der Umgebung ($in = 0$), sendet jedoch *Spikes* in die Umgebung mithilfe des Output-Neurons ($out > 0$). Dabei kann zwischen der Erzeugung einer Menge von natürlichen Zahlen oder der Erzeugung einer Sprache unterschieden werden.
Für den Fall der Erzeugung einer Menge von natürlichen Zahlen $N(\Pi)$ wird der *Spiketrain*, welcher das Output-Neuron in Abhängigkeit der Berechnung γ verlässt, unter einer Interpretation N betrachtet. Die verschiedenen Interpretationen und die entsprechenden Ergebnismengen von natürlichen Zahlen erhalten für ihren Bezeichner N bzw. $N(\Pi)$ unterschiedliche Subskripte und/oder Superskripte. Eine häufig verwendete Methode zur Interpretation und letztendlichen Berechnung des Ergebnisses ist, die vergangenen Zeitschritte zwischen mehreren vom Output-Neuron gesendeten *Spikes* zu ermitteln. Dabei leiten sich die erzeugten Ergebnismengen von natürlichen Zahlen unter den verschiedenen Interpretationen nach [PPJR06] wie folgt ab:

- Eine vom SN P-System Π erzeugte Ergebnismenge von natürlichen Zahlen unter der Interpretation N_2, nur die ersten beiden *Spikes* zu berücksichtigen:

$$N_2(\Pi) = \{t_2 - t_1 \mid \gamma \in COM(\Pi) \text{ und } st_N(\gamma) = \langle t_1, t_2, \dots \rangle\}$$

- Eine vom SN P-System Π erzeugte Ergebnismenge von natürlichen Zahlen unter der Interpretation N_k, nur die ersten $k \geq 2$ *Spikes* zu berücksichtigen:

$$N_k(\Pi) = \{t_i - t_{i-1} \mid 2 \leq i \leq k, \gamma \in COM(\Pi), st_N(\gamma) = \langle t_1, t_2, \dots \rangle \text{ und } \gamma \text{ hat mindestens } k \textit{ Spikes}\}$$

- Die Verschärfung von $N_k(\Pi)$ unter der Interpretation $N_{\underline{k}}$, nur die ersten $k \geq 2$ *Spikes* bei einem *Spiketrain* mit genau k *Spikes* zu berücksichtigen:

$$N_{\underline{k}}(\Pi) = \{t_i - t_{i-1} \mid 2 \leq i \leq k, \gamma \in COM(\Pi), st_N(\gamma) = \langle t_1, t_2, \dots \rangle \text{ und } \gamma \text{ hat genau } k \textit{ Spikes}\}$$

- Eine vom SN P-System Π erzeugte Ergebnismenge von natürlichen Zahlen unter der Interpretation N_ω, alle Berechnungen mit unendlichen *Spiketrains* zu berücksichtigen:

$$N_\omega(\Pi) = \{t_i - t_{i-1} \mid i \geq 2, \gamma \in COM(\Pi) \text{ und } st_N(\gamma) \text{ ist unendlich}\}$$

- Eine vom SN P-System Π erzeugte Ergebnismenge von natürlichen Zahlen unter der Interpretation N_{all}, alle Intervalle aller Berechnungen zu berücksichtigen:

$$N_{all}(\Pi) = \bigcup_{k \geq 2} N_k(\Pi) \cup N_\omega(\Pi)$$

- Eine vom SN P-System Π erzeugte Ergebnismenge von natürlichen Zahlen unter der Interpretation N^a, nur alternierende Intervalle aus $st_N(\gamma) = \langle t_1, t_2, \dots \rangle$ zu berücksichtigen:

$$N^a(\gamma) = \{t_{2k} - t_{2k-1} \mid k \geq 1\}$$

- Eine vom SN P-System Π erzeugte Ergebnismenge von natürlichen Zahlen unter der Interpretation N^h, nur haltende Berechnungen mit $st_N(\gamma) = \langle t_1, t_2, \dots \rangle$ und $\gamma \in HCOM(\Pi)$ zu berücksichtigen: $N^h(\Pi)$.

Die verschiedenen Mengen können miteinander kombiniert werden. Die Notation erfolgt dabei nach dem folgenden Schema: $N_\alpha^\beta(\Pi)$ mit $\beta \in \{h, a, ha\}$ oder nicht angegeben, und $\alpha \in \{\omega, all\} \cup \{k \mid k \geq 2\} \cup \{\underline{k} \mid k \geq 2\}$.
Im Gegensatz zum Berechnen des Abstandes zwischen zwei bestimmten *Spikes*, ist es auch möglich die Gesamtanzahl der gesendeten *Spikes* zu berechnen.

- Eine vom SN P-System Π erzeugte Ergebnismenge von natürlichen Zahlen unter der Interpretation N_{tot}, die Summe aller gesendeten *Spikes* zu berücksichtigen:

$$N_{tot}(\Pi) = \{\sum_{i=1}^{n} t_i \mid \gamma \in HCOM(\Pi) \text{ und } st_S(\gamma) = \langle t_1, t_2, \dots, t_n \rangle\}$$

Eine weitere Methode zur Interpretation des *Spiketrains* ist die Betrachtung der Anzahl der in die Umgebung gesendeten *Spikes* in jedem Zeitschritt. Mit dieser Interpretation L können die Ausgaben von SN P Systemen als Sprache $L(\Pi)$ betrachtet werden. Die Bezeichner für die verschiedenen Interpretationen L und die erzeugte Sprachen $L(\Pi)$ werden hierbei um Subskripte ergänzt.

- Die Betrachtung des *Spiketrains* als eine binäre Sequenz, wobei das Senden mindestens eines *Spikes* als 1 und das Senden keines *Spikes* als 0 in einem Zeitschritt interpretiert wird, wird mit $L_{bin}(\Pi)$ notiert.
- Als restriktive Form wird $L_{res}(\Pi)$ notiert, wenn das Senden keines *Spikes* in die Umgebung in einem Zeitschritt als ein Symbol des Alphabets interpretiert wird.
- Soll das Senden keines *Spikes* in die Umgebung in einem Zeitschritt als das leere Wort ε interpretiert werden, so wird $L_\lambda(\Pi)$ notiert.

3.4.2 Akzeptierender Modus

Im akzeptierenden Modus erhält das SN P-System über sein Input-Neuron eine Eingabe in einer beliebigen Kodierung. Jedoch liefert das SN P-System keine Ausgabe über ein Output-Neuron ($out = 0$). Die Eingabe, d.h. der *Spiketrain* in einer beliebigen Kodierung, wird akzeptiert, wenn das SN P-System für diese Eingabe eine Haltekonfiguration erreicht. Erreicht das SN P-System keine Haltekonfiguration, so wird die Eingabe nicht akzeptiert.
Die Eingabe kann in beliebige Kodierungen (Vgl. Interpretationen aus 3.4.1) angegeben werden. Die jedoch meist verwendeten Kodierungen sind:

- N_2 mit der Kodierung einer Zahl als Abstand zwischen zwei empfangenen *Spikes* im Input-Neuron.
- N_{tot} mit der Kodierung einer Zahl als Summe aller empfangenen *Spikes* im Input-Neuron.
- L_{bin} mit der Kodierung eines Wortes über dem Alphabet $\{0, 1\}$ als ein empfangenen *Spiketrain* st_B im Input-Neuron.
- L_{res} mit der Kodierung eines Wortes über ein Alphabet Σ als ein empfangenen *Spiketrain* st_S im Input-Neuron, wobei die Zahl 0 aus st_S dekodiert ein Element in Σ ist.
- L_{res} mit der Kodierung eines Wortes über ein Alphabet Σ als ein empfangenen *Spiketrain* st_S im Input-Neuron, wobei die Zahl 0 aus st_S dekodiert kein Element in Σ ist.

Ein SN P-System erzeugt hierbei keinen *Spiketrain* aus einer Berechnung und somit auch keine Menge. Jedoch erfordert das SN P-System eine Interpretation (nach 3.4.1) des *Spiketrains* als eine kodierte Eingabe.

3.4.3 Funktionaler Modus

Ein SN P-System im funktionalen Modus hat sowohl ein Input-Neuron, als auch ein Output-Neuron. Das SN P-System generiert in Abhängigkeit von der Eingabe, eine Ergebnismenge. Die Kodierung der Eingabe ist hierbei beliebig nach 3.4.2 und die Interpretation des resultierenden *Spiketrains* zu einer Ergebnismenge nach 3.4.1 wählbar.

3.5 BEISPIEL

Zur Veranschaulichung der Funktionsweise und der verschiedenen Anwendungsmöglichkeiten soll das folgende Beispiel dienen:

Beispiel 3.5.1. Gegeben sei der reguläre Ausdruck E mit: $E = (a + ba)(aa)^*ba$. Ziel ist es, ein SN P-System Π anzugeben, so dass $L(E) = L(\Pi)$ gilt. Ausgehend vom regulären Ausdruck E, lässt sich aus diesem nun leicht ein NEA $\mathcal{A}$ mit $L(\mathcal{A}) = L(E)$ (Abbildung 3.5.1) mittels der Thompson-Konstruktion (Definition 2.0.8) und der ε-Eliminierung (Definition 2.0.9) bilden:

$$\mathcal{A}(E) = (Q, \Sigma, q_0, \delta, F)$$

mit:

$$\begin{aligned}
Q =& \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \\
\Sigma =& \{a, b\} \\
\delta =& \{(q_0, b) \to q_1, \\
& (q_0, a) \to q_2, \\
& (q_0, a) \to q_4, \\
& (q_1, a) \to q_2, \\
& (q_1, a) \to q_4, \\
& (q_2, a) \to q_3, \\
& (q_3, a) \to q_2, \\
& (q_3, a) \to q_4, \\
& (q_4, b) \to q_5, \\
& (q_5, a) \to q_6\} \\
F =& \{q_6\}
\end{aligned}$$

Es gilt nun: $L(E) = L(\mathcal{A})$. Nun ist es möglich, aus dem entstandenen NEA ein SN P-System zu

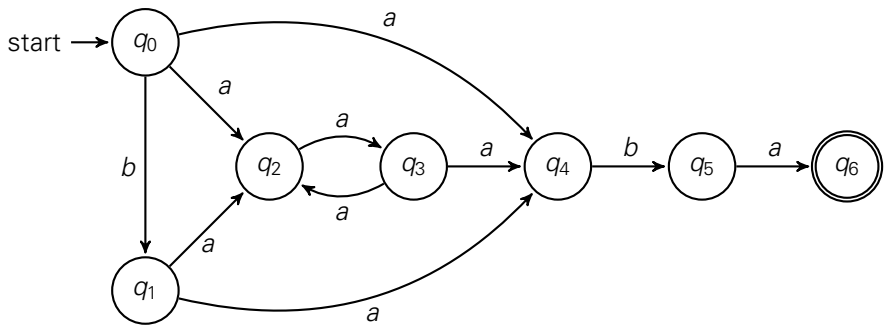


Abbildung 3.5.1: NEA $\mathcal{A}$

konstruieren. Hierbei können die Zustände des NEA in Neuronen des SN P-Systems überführt werden. Allerdings müssen für jeden Zustand noch Kontrollstrukturen in Form von zusätzlichen Neuronen hinzugefügt werden, da sonst die Eigenschaft des SN P-Systems (an alle

benachbarten Neuronen zeitgleich ein Signal zu senden) zu unkontrollierten Wortgenerierungen führen kann. Für jeden Zustand werden also zwei Neuronen konstruiert, einer für die Anwendung der Regel bzw. des Generierens eines Symbols $a \in \Sigma$ und ein Neuron zum Übermitteln eines Signals an den Folgezustand. Des Weiteren bedarf es noch eines Ausgangsneurons. Das SN P-System Π (Vgl. Abbildung 3.5.2) mit $L(\Pi) = L(E)$ ist von der Form:

$$\Pi = (O, \sigma_1, \sigma_2, ..., \sigma_{15}, syn, in, out)$$

mit:

$$\begin{aligned}
O =& \{a\} \\
\sigma_1 =& (1, \{a \to a^2; 0\, a \to a^3; 0, a \to a^5; 0\}) \\
\sigma_2 =& (0, \{a^2 \to a^3; 0, a^3 \to a^2; 0 a^5 \to a^2; 0\}) \\
\sigma_3 =& (0, \{a^2 \to a^3; 0, a^2 \to a^5; 0 a^3 \to \lambda; 0, a^5 \to \lambda; 0\}) \\
\sigma_4 =& (0, \{a^3 \to a^2; 0, a^5 \to a^5; 0\}) \\
\sigma_5 =& (0, \{a^3 \to a^4; 0, a^2 \to \lambda; 0, a^5 \to \lambda; 0\}) \\
\sigma_6 =& (0, \{a^4 \to a^2; 0\}) \\
\sigma_7 =& (0, \{a^4 \to a^3; 0, a14 \to a^5; 0\}) \\
\sigma_8 =& (0, \{a^3 \to a^2; 0, a^5 \to a^2; 0\}) \\
\sigma_9 =& (0, \{a^5 \to a^6; 0, a^2 \to \lambda; 0, a^3 \to \lambda; 0\}) \\
\sigma_{10} =& (0, \{a^6 \to a^3; 0\}) \\
\sigma_{11} =& (0, \{a^6 \to a^7; 0\}) \\
\sigma_{12} =& (0, \{a^7 \to a^2; 0\}) \\
\sigma_{13} =& (0, \{a^7 \to a^8; 0\}) \\
\sigma_{14} =& (0, \{a^8 \to \lambda; 0\}) \\
\sigma_{15} =& (0, \{a^2 \to a^2; 0, a^3 \to a^3; 0\}) \\
syn =& \{(1,2), (1,3), (1,5), (1,9), (2,15), (3,4), (3,5), (3,9), \\
& (4,15), (5,6), (5,7), (6,15), (7,8), (7,9), (8,15), (9,10), \\
& (9,11), (10,15), (11,12), (11,13), (12,15), (13,14)\} \\
in =& 0 \\
out =& 15
\end{aligned}$$

Im Folgenden werden die Neuronen $\sigma_i, i \in \mathbb{N}_0$ mit "Neuron i" angesprochen. Zu Beginn (Zeitpunkt $t = 0$) besitzt nur das Neuron 1 einen Spike. Im nächsten Zeitschritt wendet Neuron 1 nichtdeterministisch entweder die Regel $a \to a^2; 0$, die Regel $a \to a^3; 0$ oder die Regel $a \to a^5; 0$ an, was dazu führt, dass zu den Neuronen 2, 3, 5 und 9 entweder zwei Spikes (a^2), drei Spikes (a^3) oder fünf Spikes (a^5) gesendet werden. Das Ausgangsneuron (Neuron 15) hat zum jetzigen Zeitpunkt keinerlei Spikes, die es feuern könnte. Somit sendet es keinen Spike bzw. ein Wort der Länge 0 ($a^0 = \varepsilon$) in die Umgebung. Für $t = 2$ sendet Neuron 2 das von Neuron 1 empfangene Wort weiter zu Neuron 15, welches nach wie vor noch keinen Spike enthält und wiederum ein Wort der Länge 0 in die Umgebung sendet. Die Neuronen 3, 5 und 9 werden nun entsprechend des empfangenen Signals ein weiteres Wort senden, oder aber eine Forgetting Rule anwenden. Erst zum Zeitpunkt $t = 3$ sendet Neuron 15 in Abhängigkeit vom empfangenen Wort aus Neuron 2 ein Wort in die Umgebung mit der Länge 2 oder 3. Zur gleichen Zeit feuert das Neuron 4 bzw. 6 oder 10 ein Wort zum Ausgangsneuron 15, um im nächsten Zeitpunkt nahtlos das Wort in die Umgebung senden zu können. Erst, wenn sich Neuron 7 nach $2n$ Schritten nichtdeterministisch entscheidet, die Regel $a^4 \to a^5; 0$ anzuwenden, kann das Neuron 9 ein Wort der Länge 6 zu den Neuronen 10 und 11 senden. Nach insgesamt $3 + i + 2m + 1$ mit $i \in \{0, 1\}$ und $m \in \mathbb{N}_0$ Zeitschritten, erreicht das System eine Haltekonfiguration C_H mit der Berechnung $\gamma = C_0 \Rightarrow \cdots \Rightarrow C_H$ und stoppt mit den an die Umgebung gesendeten Spiketrain $st_S(\gamma)$. Wobei gilt:

$$L_\lambda(\Pi) = \{st_S(\gamma) \mid \gamma \in HCOM(\Pi)\}$$

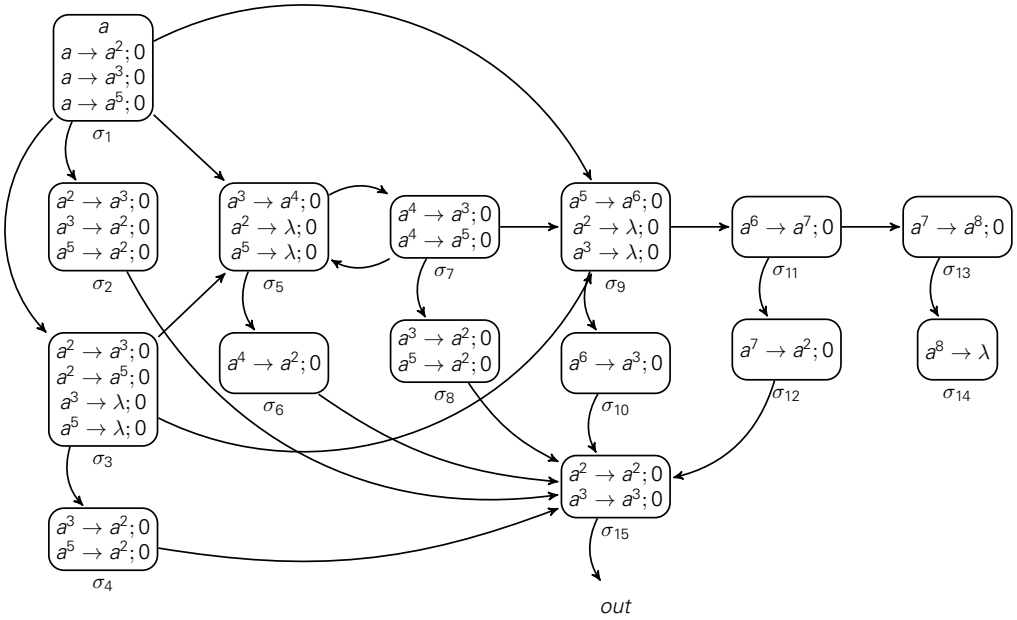


Abbildung 3.5.2: SN P-System Π

Es bedarf nun noch einer Übersetzung- bzw. Dekodierungsfunktion, so dass die gleiche Sprache wie $L(\mathcal{A})$ abgebildet wird. Dazu werden für eine Sprache $L(\Pi)$, den Symbolen $a, b \in \Sigma$ und einen Spiketrain $st_S(\gamma) = \langle t_1, \ldots, t_n \rangle$ mit $t_i \in \mathbb{N}_0$ und $i, n \in \mathbb{N}_1$ definiert:

$$trans(L_\lambda(\Pi)) = \bigcup_{st_S(\gamma) \in L_\lambda(\Pi)} dec(t_1) \cdot \ldots \cdot dec(t_n) \qquad dec(t_i) = \begin{cases} \varepsilon & \text{falls } t_i = 0 \\ a & \text{falls } t_i = 2 \\ b & \text{falls } t_i = 3 \end{cases}$$

Mittels der Anwendung der Übersetzung- und Dekodierungsfunktionen gilt nun:

$$\begin{aligned} trans(L_\lambda(\Pi)) &= L(\mathcal{A}) \\ &= L(E) \end{aligned}$$

Ein SN P-System für den regulären Ausdruck E wurde somit gefunden.
Tabelle 3.5.1 zeigt die Generierung eines Wortes $w = aba$, für welches gilt:
$aba = dec(t_1) \cdot \ldots \cdot dec(t_n)$ mit $st_S(\gamma) = \langle t_1, \ldots, t_n \rangle$ und $\gamma \in HCOM(\Pi)$. Hierbei sind die Zellen der Tabelle in zwei Zeilen aufgeteilt. Die obere Zeile gibt das derzeitige im Neuron i enthaltene Wort w_j zum Zeitpunkt t an wobei der Index j anzeigt, von welchem Neuron das Wort gesendet wurde, und die untere Zeile gibt die zu verwendete Regel des Neurons i zu diesem Zeitpunkt an. Zum Zeitpunkt $t = 5$ sind in keinen Neuronen noch Wörter enthalten, daher kann keine Regel mehr angewendet werden und es wurde eine Haltekonfiguration C_H erreicht. Das System stoppt und der vom Neuron 15 gelieferte Spiketrain bildet eine Lösung des Systems.

Tabelle 3.5.1: Berechnung des Wortes w

$\sigma_i \backslash t$	0	1	2	3	4	5
1	a_1	–	–	–	–	–
	–	$a \to a^5; 0$	–	–	–	–
2	–	a_1^5	–	–	–	–
	–	–	$a^5 \to a^2; 0$	–	–	–
3	–	a_1^5	–	–	–	–
	–	–	$a^5 \to \lambda; 0$	–	–	–
4	–	–	–	–	–	–
	–	–	–	–	–	–
5	–	a_1^5	–	–	–	–
	–	–	$a^5 \to \lambda; 0$	–	–	–
6	–	–	–	–	–	–
	–	–	–	–	–	–
7	–	–	–	–	–	–
	–	–	–	–	–	–
8	–	–	–	–	–	–
	–	–	–	–	–	–
9	–	a_1^5	–	–	–	–
	–	–	$a^5 \to a^6; 0$	–	–	–
10	–	–	a_9^6	–	–	–
	–	–	–	$a^6 \to a^3; 0$	–	–
11	–	–	a_9^6	–	–	–
	–	–	–	$a^6 \to a^7; 0$	–	–
12	–	–	–	a_{11}^7	–	–
	–	–	–	–	$a^7 \to a^2; 0$	–
13	–	–	–	a_{11}^7	–	–
	–	–	–	–	$a^7 \to a^8; 0$	–
14	–	–	–	–	a_{13}^8	–
	–	–	–	–	–	$a^8 \to \lambda; 0$
15	–	–	a_2^2	a_{10}^3	a_{12}^2	–
	–	–	–	$a^2 \to a^2; 0$	$a^3 \to a^3; 0$	$a^2 \to a^2; 0$

4 MODIFIKATIONEN VON SN P-SYSTEMEN

SN P-Systeme können nach verschiedenen Schwerpunkten modifiziert werden. Dies hat zur Folge, dass entsprechend der Regeln, des Verhaltens und des spezifischen Aufbaus der SN P-Systeme, unterschiedliche Ergebnismengen von natürlichen Zahlen oder formale Sprachen generiert und/oder akzeptiert werden können. In dieser Arbeit werden die SN P-Systeme im Bezug auf die Modifikation der Regeln, des Verhaltens, der Form und der zusätzlichen Komponenten unterteilt. Entsprechend dieser Modifikationen und Eigenschaften eines SN P-Systems, können die SN P-Systeme in Klassen bezüglich ihrer resultierenden Ergebnismengen unterschieden werden. Eine solche für SN P-Systemen spezifizierte Klasse ist ein Mengensystem, dessen Bezeichner sich aus zwei Hauptbestandteilen zusammensetzt. Zum einen die zu erzeugende bzw. akzeptierende (Vgl. Kapitel 3.4) Ergebnismenge *Res* und zum anderen einen allgemeinen Bezeichner $Mb_1\,SN^{Mb_2}_{Modus}P$ eines SN P-Systems, das die Menge *Res* erzeugt bzw. akzeptiert. Für den Bezeichner einer solchen Klasse der SN P-Systeme wird in dieser Arbeit die folgende Form eingeführt:

$$ResMb_1\,SN^{Mb_2}_{Modus}P$$

wobei gilt:

- *Res* ist die Ergebnismenge nach der Interpretation eines *Spikestrains* gemäß Kapitel 3.4.
- $Mb_1\,SN^{Mb_2}_{Modus}P$ ist der konkrete Bezeichner für das zu betrachtende SN P-System, wobei gilt:
  - Mb_1 und Mb_2 sind Modifikationsbezeichner. Je nach Modifikation des SN P-Systems werden Präfixe (Mb_1) oder Superskripts (Mb_2) zur eindeutigen Bezeichnung eingesetzt. Sind keine Modifikationsbezeichner für das entsprechende SN P-System vorgesehen, so entfällt die Angabe von Mb_1 und/oder Mb_2.
  - $Modus \in \{gen, acc, func\}$ ist der Bezeichner des zu verwendeten Modus im Bezug auf die Ergebnismenge *Res* des SN P-Systems. Das SN P-System kann sich hierbei im generativen (*gen*), im akzeptierenden (*acc*) oder im funktionalen (*func*) Modus befinden. Standardmäßig befindet sich ein SN P-System im generativen Modus, daher kann für $Modus = gen$ die Angabe von *Modus* entfallen.

Eine Unterklasse kann gebildet werden, wenn für die Klasse $ResMb_1\,SN^{Mb_2}_{Modus}P$ Einschränkungen festgelegt werden. Dabei wird hinter der Klasse eine Aufzählung von beliebig gewählten Eigenschaften $prop_i$ notiert:

$$ResMb_1\,SN^{Mb_2}_{Modus}P_m(prop_1, prop_2, \ldots, prop_n)$$

Dabei ist eine Eigenschaft $prop_i$ eine der folgenden Bezeichner:

- $rule_k$ ist die maximale Anzahl k der Regeln in den Neuronen. Eine Eigenschaft $rule_k$ ist stärker eingeschränkt als $rule_{k'}$, wenn $k < k'$ gilt.
- $cons_p$ ist die maximale Anzahl p der zur konsumierenden *Spikes*. Eine Eigenschaft $cons_p$ ist stärker eingeschränkte als $cons_{p'}$, wenn $p < p'$ gilt.
- $forg_f$ ist die maximale Anzahl f der zu löschenden *Spikes* in einer *Forgetting Rule* in den Neuronen. Eine Eigenschaft $forg_f$ ist stärker eingeschränkt als $forg_{f'}$, wenn $f < f'$ gilt.
- $prod_q$ ist die maximale Anzahl q der zu produzierenden *Spikes*. Eine Eigenschaft $prod_q$ ist stärker eingeschränkt als $prod_{q'}$, wenn $q < q'$ gilt.

Gesondert wird die Einschränkung der maximalen Anzahl der Neuronen m angegeben. Eine Einschränkung m' ist stärker als m, wenn $m' < m$ gilt. Sind Eigenschaften nicht beschränkt, so entfällt die Angabe der eingeschränkten Eigenschaft $prop_i$ in der Aufzählung. Sind Eigenschaften beschränkt, ist aber die Stufe der Einschränkung unbestimmt, so wird für eine Eigenschaft $prop_i$ ein $*$ statt einer quantitativen Einschränkung (k, p, f oder q) notiert. Jede quantitative Einschränkung ist stärker eingeschränkt als $*$.

Lemma 4.0.1. $ResMb_1 SN^{Mb_2}_{Modus} P_{m'}(prop_{1'}, prop_{2'}, \ldots, prop_{n'}) \subseteq ResMb_1 SN^{Mb_2}_{Modus} P_m(prop_1, prop_2, \ldots, prop_n) \subset ResMb_1 SN^{Mb_2}_{Modus} P$ für alle $m' \leq m \leq 1$ und $prop_{i'}$ ist stärker oder genauso stark eingeschränkt wie $prop_i$.

Lemma 4.0.1 folgt unmittelbar aus [IPY06]. Eine Klasse der SN P-Systeme $A' = ResMb_1 SN^{Mb_2}_{Modus} P_{m'}(prop_{1'}, prop_{2'}, \ldots, prop_{n'})$ ist nur dann eine Unterklasse von $A = ResMb_1 SN^{Mb_2}_{Modus} P_m(prop_1, prop_2, \ldots, prop_n)$, Wenn jede Eigenschaft aus A' mindestens genauso stark eingeschränkt ist, wie die entsprechende Eigenschaft aus A.

4.1 MODIFIKATION BEZÜGLICH DER REGELN

SN P-Systeme können bezüglich ihrer Regeln in den einzelnen Neuronen modifiziert werden. Eine Methode ist die Darstellung der einzelnen Regeln entsprechend einer speziellen Notation. Dazu betrachte man eine Regel r nach Definition 3.1.1 für den Fall einer *Firing Rule* $r = E/v \rightarrow w; d$. Es haben sich hierbei die folgenden Notationen durchgesetzt:

Notation 4.1.1. Wie bereits in 3.2 erwähnt, kann die Angabe von E in einer Regel r entfallen, wenn $L(E) = \{v\}$ gilt. Dabei wird vereinfacht $r = v \rightarrow w; d$ notiert.

Notation 4.1.2. Sei r eine Regel und d die Verzögerung der Regel r. Wenn $d = 0$ gilt, so kann die Angabe von d entfallen und es wird vereinfacht $r = E/v \rightarrow w$ notiert.

Des Weiteren können die Regeln (und deren Neuronen) in verschiedene Charakteristika unterschieden werden.

1. Eine Regel r heißt *beschränkt*, wenn sie von der Form $a^j/a^j \rightarrow w; d$ oder von der Form $a^k \rightarrow \lambda$ ist, mit $j \leq i$. Ein Neuron, das nur beschränkte Regeln enthält, ist ebenfalls beschränkt. Besteht ein SN P-System nur aus beschränkten Neuronen, so ist das SN P-System ebenfalls beschränkt.
2. Eine Regel r heißt *unbeschränkt*, wenn sie von der Form $E/a^j \rightarrow w; d$ ist, mit $L(E)$ ist unendlich. Ein Neuron, das nur unbeschränkte Regeln enthält, ist ebenfalls unbeschränkt. Besteht ein SN P-System nur aus unbeschränkten Neuronen, so ist es ebenfalls unbeschränkt.

Für jeden anderen Fall nennt man das Neuron bzw. das SN P-System *allgemein*. Die Regeln können zudem bezüglich der Anzahl der zu sendenden *Spikes* modifiziert werden. Im Allgemeinen unterscheidet man hier zwischen restriktiven (standardisierten) und erweiterten Regeln.

4.1.1 SN P-Systeme mit restriktiven Regeln

In [IPY06] wurde das SN P-System mit restriktiven Regeln als das Standard-SN P-System vorgestellt. Die Idee hinter diesem SN P-System ist, dass jedes Neuron maximal einen *Spike* weitersenden kann. Dieses SN P-System stellt eine Grundlage für weitere Modifikationen dar. Gegeben sei ein SN P-System Π nach 3.1.1. Die Regeln werden dabei wie folgt modifiziert:

- Für eine *Firing Rule* $r = E/v \rightarrow w; d$ gilt: $|w| = 1$. Verkürzt kann daher $r = E/v \rightarrow a; d$ mit $a \in O$ geschrieben werden.
- Für eine *Forgetting Rule* r gilt nach wie vor: $r = E/u \rightarrow \lambda$.

Aufgrund dessen, dass das SN P-System mit restriktiven Regeln als Standard-SN P-System gilt, erhält dieses System keine Namenserweiterungen. Wird also ein System nur als SN P-System bezeichnet, so ist davon auszugehen, dass es sich um ein System mit restriktiven Regeln handelt. Die Neuronen eines solchen SN P-Systems können jeweils nur einen einzigen *Spike* pro Regel senden. Somit kann auch das Output-Neuron nur maximal einen *Spike* pro Zeitschritt in die Umgebung senden. Dies hat zur Folge, dass man entweder nur einen gesendeten *Spike* oder keinen gesendeten *Spike* für den *Spiketrain* beobachten kann. Zur Interpretation des *Spiketrains* bleiben somit nur wenige Möglichkeiten. Die erste, auch gebräuchlichste, Möglichkeit ist das Messen der vergangenen Zeitschritte zwischen genau zwei *Spikes* (Vgl. $N_2(\Pi)$ aus 3.4.1). Konventionell wird daher die Klasse der SN P-Systeme, welche die Ergebnismengen über $N_2(\Pi)$ beinhaltet, mit $N_2SNP_m(rule_k, cons_p, forg_f)$ oder, aufgrund des häufigen Gebrauchs, auch mit $Spik_2P_m(rule_k, cons_p, forg_f)$ nach [IPY06] notiert. Die zweite Möglichkeit ist das Interpretieren des *Spikestrains* als eine binäre Sequenz (Vgl. $L_{bin}(\Pi)$ aus 3.4.1) mit $L_{bin}SNP_m(rule_k, cons_p, forg_f)$.

Theorem 4.1.1. $Spik_2P_*(rule_k, cons_p, forg_f) = NRE$
für alle $k \geq 2, p \geq 3, f \geq 3$.

Der Beweis erfolgte bereits in [IPY06]. Die Idee ist hierbei, dass eine Registermaschine mit einem restriktiven SN P-System simuliert werden soll. Dabei ist von einer Registermaschine bereits bekannt, dass sie Turing-berechenbare Funktionen berechnen kann. Ein SN P-System wird dabei nach den drei Hauptmodulen einer Registermaschine konstruiert, so dass eine Additionsinstruktion, eine Subtraktionsinstruktion und eine Halteinstruktion simuliert werden. Aus Theorem 4.1.1 folgt, dass ein restriktives SN P-System Turing-berechenbare Funktionen berechnen kann. Als nächstes kann nun untersucht werden, ob restriktive SN P-Systeme rekursiv aufzählbare Sprachen (z.B. L_{bin}) generieren. Dazu dient das Theorem aus [Che+07].

Theorem 4.1.2. Es gibt Sprachen $L \in FIN$ (zum Beispiel $\{0^k, 10^j\}$ für beliebige aber feste $k \geq 1, j \geq 0$), die nicht von einem restriktiven SN P-System Π mit $L(\Pi) \in L_{bin}SNP$ generiert werden können.

Der Beweis erfolgte bereits in [Che+07] mithilfe eines Gegenbeispiels. Um ein Wort $10^j \in \{0^k, 10^j\}$ generieren zu können, muss sich bereits in der Startkonfiguration ein *Spike* im Output-Neuron befinden, welcher im nächsten Zeitschritt mittels einer Regel $r_1 = E/v \rightarrow w; d$ in die Umgebung gesendet wird. Allerdings kann nun kein Wort $0^k \in \{0^k, 10^j\}$ mehr generiert werden, da das Output-Neuron nun eine *Forgetting Rule* $r_2 = u \rightarrow \lambda$ benötigt, um den vorhandenen *Spike* in der Startkonfiguration zu löschen und keinen *Spike* im nächsten Zeitschritt in die Umgebung senden zu können. Nach Definition 3.1.1 gilt jedoch $u \notin L(E)$. Somit können nicht alle Wörter aus $\{0^k, 10^j\}$ von einem restriktiven SN P-System generiert werden. Aus Theorem 4.1.2 folgt somit, dass restriktive SN P-System bezüglich L_{bin} keine rekursiv aufzählbaren Sprachen generieren können, sondern nur eine Teilmenge von endlichen Sprachen ($L_{bin}SNP \subset FIN$).

Beispiel 4.1.1. Gegeben sei das folgende SN P-System Π_2 mit $\Pi_2 = (\{a\}, \sigma_1, \ldots, \sigma_{n+2}, syn, in, out)$:

$$\begin{aligned}
\sigma_1 &= (0, \{a \to a; 0, a^2 \to \lambda\}) \\
\sigma_2 = \cdots = \sigma_n &= (0, \{a \to a; 0\}) \\
\sigma_{n+1} &= (0, \{a \to a; 0, a \to a; 1\}) \\
\sigma_{n+2} &= (3, \{a^3 \to a; 0, a^2 \to a; 0, a \to \lambda\}) \\
syn &= \{(1,2), (n-2, n-1), (n-2, n+1), (n-1, n), (n-1, n+2), (n+1, n+2), \\
&\qquad (n,1), (n+2,1)\} \cup \bigcup_{i \geq 2}^{n-2} \{(i, i+1)\} \\
in &= 0 \\
out &= n+2
\end{aligned}$$

Im Folgenden werden die Neuronen entsprechend ihrer Darstellung in Abbildung 4.1.1

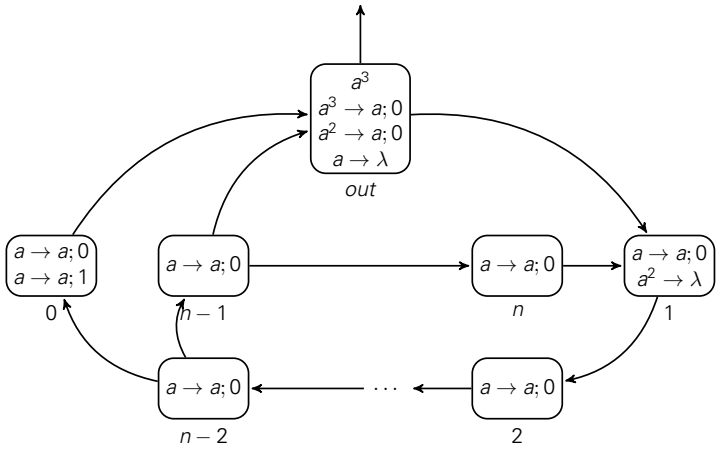


Abbildung 4.1.1: SN P-System Π_2 mit restriktiven Regeln nach [IPY06]

angesprochen. Dabei stellt das Neuron $n+1$ Neuron 0 und das Neuron $n+2$ Neuron *out* dar. Zum Zeitpunkt 0 besitzt nur das Neuron *out* drei *Spikes*, wodurch im Zeitpunkt sofort ein *Spike* in die Umgebung gesendet wird. Da für dieses SN P-System der Zeitabstand zwischen den ersten beiden *Spikes* gemessen wird, beginnt die Zählung im Zeitpunkt 2, wenn das erste Mal kein *Spike* nach dem zuvor gesendeten *Spike* in die Umgebung gesendet wird. Im Zeitpunkt 1 erhält ebenfalls das Neuron 1 einen *Spike* und kann somit im Zeitpunkt 2 einen *Spike* zum Neuron 2 senden. Pro Zeitschritt wird nun ein *Spike* vom Neuron 2 entlang nach n Zeiteinheiten zum Neuron $n-2$ gesendet. Im nächsten Zeitschritt erhalten nun Neuron 0 und Neuron $n-1$ einen *Spike*, woraufhin sich Neuron 0 im nächsten Zeitschritt nichtdeterministisch für eine Regel entscheidet. Wird die Regel $a \to a; 1$ angewendet, so wird im nächsten Zeitschritt der einzige empfangene *Spike* vom Neuron *out* verworfen. Erhält Neuron 1 erneut einen *Spike*, so sind bereits $n+1$ Zeitschritte vergangen nachdem der erste *Spike* in die Umgebung gesendet wurde. Entscheidet sich Neuron 0 nach i Zyklen die Regel $a \to a; 0$ anzuwenden, so wird Neuron *out* im nächsten Zeitschritt den zweiten *Spike* in die Umgebung senden und die Berechnung hält. Das SN P-System Π ermittelt somit ein Vielfaches von n.

$$N_2(\Pi_2) = \{ni \mid n \geq 3 \text{ und } i \in \mathbb{N}_1\} \in Spik_2P_{n+2}(rule_3, cons_3, forg_2).$$

4.1.2 Spiking Neural P-Systeme mit erweiterten Regeln

Eine weitere Modifikation bezüglich der Regeln wird in [Che+08] vorgestellt. Aufgrund der Tatsache, dass SN P-Systeme mit restriktiven Regeln sowohl keine formalen Sprachen aus

Klassen über *FIN* hinaus, als auch nicht alle endlichen Sprachen darstellen können, bedarf es eines Systems, mit dem formale Sprachen aus *RE* erzeugt werden können. Man beschränkt sich hierbei nicht mehr darauf, dass pro Regel, also pro Neuron in einem Zeitschritt, nicht nur ein *Spike* sondern beliebig viele *Spikes* in Abhängigkeit von abgeschwächten Restriktionen gesendet werden dürfen. Die Regeln werden dabei wie folgt modifiziert:

- Für eine *Firing Rule* $r = E/v \rightarrow w; d$ gilt: $|v| \geq 1$ und $|v| \geq |w| \geq 0$.
- Eine *Forgetting Rule* ist dementsprechend eine spezielle Form der *Firing Rule* mit $|w| = 0$. Des Weiteren kann eine *Forgetting Rule* weiterhin mit $E/u \rightarrow \lambda$ notiert werden, sofern $|w| = 0$ und $d = 0$ gilt.

Ein SN P-System mit erweiterten Regeln wird als *erweitertes* SN P-System bezeichnet. Die Klasse der SN P-Systeme erhält für seinen Bezeichner $Mb_2 = e$. Ein *erweitetes* SN P-System erweitert den Berechnungsumfang von SN P-Systemen mit restriktiven Regeln um die Fähigkeit, formale Sprachen aus den Klassen über *FIN* hinaus mittels der vereinfachten Form $L_\lambda(\Pi)$ oder der restriktiven Form $L_{res}(\Pi)$ darzustellen. Zunächst wird allerdings noch untersucht, ob *erweiterte* SN P-Systeme nach wie vor eine Menge von natürlichen Zahlen berechnen können.

Theorem 4.1.3. $N_2SN^eP_2(rule_4, cons_5, prod_2) = NRE$.

Der Beweis erfolgte bereits in [Che+08]. Die Idee ist auch hierbei wie im Theorem 4.1.1 die Simulation einer Registermaschine bezüglich ihrer drei wichtigsten Instruktionen: Addition, Subtraktion und Halten. Eine Konstruktion eines *erweiterten* SN P-Systems verhält sich äquivalent.
Es ist somit möglich, Turing-berechenbare Funktionen mit *erweiterten* SN P-Systemen zu berechnen. Des Weiteren kann nun untersucht werden, welche Sprachen mit einem *erweiterten* SN P-System im Gegensatz zu einem SN P-System mit restriktiven Regeln berechnet werden können.

Theorem 4.1.4. Sei L eine reguläre Sprache über dem Alphabet Σ und s_0 ein weiteres Symbol mit $s_0 \notin \Sigma$. So gilt: $\{s_0\} \cdot L \in L_{res}SN^eP_4(rule_*, cons_*, prod_*)$. Es gilt ebenfalls:
$L \cdot \{s_0\} \in L_{res}SN^eP_3(rule_*, cons_*, prod_*)$.

Der Beweis erfolgte bereits in [Che+08]. Die Beweisidee ist dabei die Konstruktion eines *erweiterten* SN P-Systems und einer regulären Grammatik G, so dass $L = L(G)$ gilt. Ob das Sondersymbol s_0 auf irgendeine Weise entfallen kann, ist laut [Che+08] unbekannt. Die Verwendung von $L_\lambda(\Pi)$ bietet allerdings die Möglichkeit, ein solches Sondersymbol zu berücksichtigen und geht in der Berechnungsstärke über die Klasse der regulären Sprachen hinaus.

Theorem 4.1.5. $L_\lambda SN^eP_*(rule_*, cons_*, prod_*) = RE$.

Der Beweis erfolgte in [Che+08] über die Konstruktion zweier Registermaschinen $\mathcal{M}_0$ und $\mathcal{M}_1$. Für die Beweisidee wird m für die Anzahl der Symbole in einem Alphabet $\Sigma = \{s_1, \ldots, s_m\}$ und $val_m(w)$ für den Wert zur Basis $m+1$ eines Wortes $w \in \Sigma^*$ notiert. Durch die Verwendung der Basis $m+1$ wird jedes Wort eineindeutig einer natürlichen Zahl zugeordnet. Mit $val_m(L)$ wird die Menge von allen $val_m(w)$ mit $w \in L$ notiert. Des Weiteren gilt: $L \in RE$, wenn $val_m(L) \in NRE$. Eine Menge von natürlichen Zahlen (hier $val_m(L)$) ist rekursiv aufzählbar, wenn es eine deterministische Registermaschine $\mathcal{M}_1$ gibt, die genau diese Menge akzeptiert. Die Registermaschine $\mathcal{M}_0$ berechnet das als Zahl kodierte Wort aus den einzelnen als Zahlen kodierten Symbolen. Registermaschine $\mathcal{M}_0$ und $\mathcal{M}_1$ können nun in ein *erweitertes* SN P-System überführt werden und dadurch wird gezeigt, dass mit *erweiterten* SN P-Systemen rekursiv aufzählbare Sprache generiert werden können.

Zur Veranschaulichung der Generierung einer formalen Sprache soll das folgende Beispiel aus [Che+08] dienen.

Beispiel 4.1.2. Gegeben sei das *erweiterte* SN P-System Π_3 mit $\Pi_3 = (\{a\}, \sigma_1, \sigma_2, \sigma_3, syn, in, out)$:

$$\begin{aligned}
\sigma_1 &= (6, \{a^2(a^4)^+/a^4 \to a^4, a^2 \to a^2, a^6 \to a^3\}) \\
\sigma_2 &= (0, \{a^4 \to a^4, a^2 \to a^2\}) \\
\sigma_3 &= (0, \{a^2(a^4)^+/a^4 \to a^4, a^2 \to a^2\}) \\
syn &= \{(1,2), (1,3), (2,3), (3,1)\} \\
in &= 0 \\
out &= 3
\end{aligned}$$

Die Neuronen werden entsprechend nach ihrer Darstellung in Abbildung 4.1.2 angesprochen.

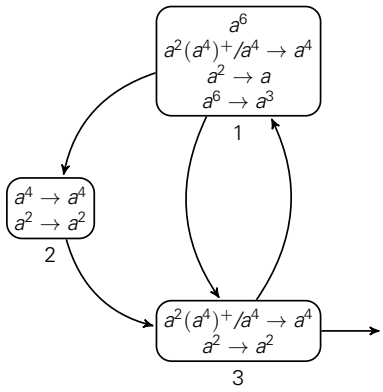


Abbildung 4.1.2: *erweitertes* SN P-System Π_3 nach [Che+08]

Zum Zeitpunkt 0 besitzt nur das Neuron 1 sechs *Spikes*. Sobald sich Neuron 1 entscheidet, die Regel $a^6 \to a^3$ anzuwenden, hält die Berechnung. Wendet Neuron 1 hingegen die Regel $a^2(a^4)^+/a^4 \to a^4$ an, so erhalten die Neuronen 2 und 3 jeweils vier *Spikes*. Geht man davon aus, dass die Symbole der Wörter der zu generierenden Sprache durch die *Spike*-Anzahlen 2 (s_2) und 4 (s_4) repräsentiert werden, so kann man sehen, dass sich im nächsten Zeitschritt innerhalb des Neurons 3 die Anzahl für den Repräsentanten von 4 verdoppelt hat. im Neuron 3 befinden sich nun ein mal 2 und zwei mal 4 (= 10) *Spikes*, die in den nächsten Zeitschritten in die Umgebung gesendet werden. Sobald Neuron 3 zwei *Spikes* in die Umgebung gesendet hat, besitzt Neuron 1 wieder ein mal 2 und zwei mal 4 *Spikes*. Es ist nun zu erkennen, dass für jeden weiteren Zyklus, die Anzahl der 4 *Spikes* verdoppelt wird. Bis Neuron 1 die Regel $a^6 \to a^3$ anwendet und das System hält. Das *erweiterte* SN P-System Π_3 erzeugt somit die Sprache:

$$L_\lambda(\Pi_3) = \{s_4^2 s_2 s_4^{2^2} \cdots s_4^{2^n} s_2 \mid n \geq 1\} \in L_\lambda SN^e P_3(rule_3, cons_6, prod_4)$$

4.2 MODIFIKATION BEZÜGLICH DER FORM

Neben den Änderungen an der formalen Struktur eines SN P-Systems, können auch Modifikationen im Bezug auf die Darstellungsform angewendet werden. Spezifische Darstellungsformen können dabei helfen, ein SN P-System dem biologischem Nervensystem nahe zu bringen. Des Weiteren helfen bestimmte einheitliche Formen, spezifische modifizierte SN P-Systeme miteinander zu vergleichen.

4.2.1 Homogene Spiking Neural P-Systeme

Mit der Darstellungsform von homogenen SN P-Systemen wird versucht, eine einheitliche Darstellung aller Neuronen in einem SN P-System zu erhalten. Dies hat den Vorteil, dass der

Bezug zum biologischen Nervensystem näher gebracht werden kann. Beispielsweise besteht das (menschliche) Gehirn aus Milliarden von Nervenzellen. Die meisten dieser Nervenzellen sind von Natur aus so konstruiert, dass sie gleiche Aufgaben erfüllen, demzufolge über identische Regeln verfügen. Das Auftreten von identischen Regeln in den verschiedenen Neuronen soll mit homogenen SN P-Systemen simuliert werden.
Ein homogenes SN P-System (verkürzt als HSN P-System bezeichnet) ist nach [ZZP09] ein SN P-System $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$ nach Definition 3.1.1 mit den folgenden Bedingungen:

- Ein SN P-System heißt homogen, wenn für alle Neuronen $\sigma_i = (n_i, R_i)$ mit $1 \leq i \leq m$ gilt: $R_1 = R_2 = \ldots = R_m$.
- Ein SN P-System heißt semi-homogen, wenn alle Neuronen außer einem, die Identische Menge der Regeln besitzen.

Allerdings muss nun untersucht werden, ob ein solches HSN P-System, trotz seiner Beschränkung im Gegensatz zu einem SN P-System mit restriktiven Regeln, immer noch in der Lage ist, Turing-berechenbare Funktionen zu berechnen.

Theorem 4.2.1. $N_2HSNP = NRE$.

In [ZZP09] wurde das Theorem 4.2.1 über die Klasse $N_2HSNP_*(rule_7, cons_7, forg_4)$ bewiesen. Die Idee ist hierbei, zunächst vereinfacht eine Registermaschine mit gewichteten HSN P-Systemen zu simulieren. Die gewichteten Synapsen werden anschließend durch weitere Neuronen ersetzt. Um die homogene Form zu erhalten, muss eine bestimmte Anzahl an Neuronen ergänzt werden. Die Anzahl ist abhängig von den gewichteten Synapsen, die durch homogene Neuronen ersetzt wurden.

Beispiel 4.2.1. Gegeben sei das einfache HSN P-System Π mit $\Pi_4 = (\{a\}, \sigma_1, \sigma_2, \sigma_3, syn, in, out)$, das alle Elemente aus $\mathbb{N}_1$ generieren soll:

$$\begin{aligned}
\sigma_1 &= (1, R)\\
\sigma_2 &= (0, R)\\
\sigma_3 &= (1, R)\\
R &= \{a \to a; 0, a \to a; 1, a^2 \to \lambda\}\\
syn &= \{(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)\}\\
in &= 0\\
out &= 3
\end{aligned}$$

Die Neuronen werden auch hier wieder entsprechend ihrer Darstellung in 4.2.1 mit Neuron i

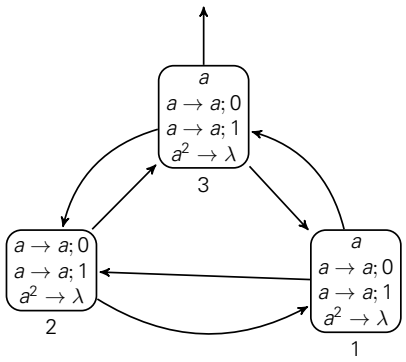


Abbildung 4.2.1: HSN P-System Π_4

angesprochen. Zum Zeitpunkt 0 besitzen nur Neuron 1 und Neuron 2 einen *Spike*. Beide Neuronen können sich nun jeweils nichtdeterministisch für die Regel $a \to a; 0$ oder für die Regel

$a \to a; 1$ entscheiden. Daraus entstehen nun vier Möglichkeiten. Entscheiden sich sowohl Neuron 1 als auch Neuron 0 die Regel $a \to a; 0$ anzuwenden, so wird im nächsten Zeitschritt ein *Spike* in die Umgebung und zum Neuron 3 gesendet, welches wiederum im darauffolgenden Zeitschritt einen weiteren *Spike* in die Umgebung senden kann, so dass mit der Interpretation N_2 eine Lösung (1) gefunden wurde. Das gleiche Ergebnis entsteht, wenn sich beide Neuronen für die Regel $a \to a; 1$ entscheiden. Das gleiche Ergebnis ist zu erkennen, wenn Neuron 1 sofort feuert und Neuron 3 erst nach einem weiteren Zeitschritt feuert. Nur wenn Neuron 3 sich für die Regel $a \to a; 0$ und Neuron 1 sich für die Regel $a \to a; 1$ entscheiden, so kann ein anderes Ergebnis erzielt werden. Somit haben Neuron 1 und Neuron 2 jeweils einen *Spike*. Feuern beide Neuronen immer zur gleichen Zeit einen *Spike*, so erhält Neuron 3 zwei *Spikes* womit im nächsten Zeitschritt kein *Spike* in die Umgebung aufgrund der Regel $a^2 \to \lambda$ gesendet wird. Erst wenn Neuron 1 und Neuron 2 nicht gleichzeitig einen *Spike* senden, kann durch die Anwendung der Regel $a \to a; 0$ im nächsten Zeitschritt eine Lösung geliefert werden. Die Ergebnismenge des HSN P-Systems Π_4 ist somit:

$$N_2(\Pi_4) = \{1, 2, \ldots, n \mid n \in \mathbb{N}_1\} \in N_2HSNP_3(rule_3, cons_2, forg_2)$$

4.2.2 Zeitunabhängige Spiking Neural P-Systeme

Eine weitere Form von SN P-Systemen sind die zeitunabhängigen SN P-Systeme. Unabhängig von der möglichen Verzögerung in den Regeln, liefern diese Systeme immer das gleiche Ergebnis. In einem biologischen Nervensystem werden die Signale nach einer gewissen Zeit gesendet, jedoch nicht sofort nach der Anwendung ($d = 0$). Zeitgesteuerte SN P-Systeme betrachten diese Form der Nervensysteme. Durch biologische Einflüsse können jedoch die Verzögerungen zum Senden der Signale stark verändert werden. Um die Auswirkungen solcher Einflüsse zu verringern bzw. vermeiden, werden zeitunabhängige SN P-Systeme eingeführt. Um *zeitunabhängige* SN P-Systeme genauer zu erläutern, wird zunächst das SN P-System Π aus Definition 3.1.1 zu einem *zeitgesteuertem* SN P-System nach [PZZ11] modifiziert:

$$\Pi(e) = (O, \sigma_1, \ldots, \sigma_m, syn, in, out, e)$$

- $e : R_1 \cup \ldots \cup R_m \mapsto \mathbb{N}_1$ ist eine Abbildung, die einer Regel r eine Verzögerung d zuordnet.

Aus dieser Beschreibung kann nun die Bedingung für ein *zeitunabhängiges* SN P-System formal erfasst werden:

- Ein SN P-System heißt *zeitunabhängig*, wenn jedes *zeitgesteuerte* SN P-System in der Menge $\{\Pi(e) \mid e : R_1 \cup \ldots R_m \mapsto \mathbb{N}_1\}$ die gleiche Lösungsmenge produziert.

Für ein solches *zeitunabhängiges* SN P-System erhält die Klasse der SN P-System den Modifikationszusatz *free*. Aufgrund der Änderung im Verzögerungsverhalten kann für die Berechnung einer Menge von natürlichen Zahlen nicht mehr der Abstand zwischen zwei *Spikes* (N_k) berechnet werden. Stattdessen berechnet man nun die Gesamtanzahl der gesendeten *Spikes* (N_{tot}). Die Untersuchung der Berechnungsstärke eines *zeitunabhängigen* SN P-Systems wurde bereits in [PZZ11] durchgeführt.

Theorem 4.2.2. $N_{tot}SN^{free}P = NRE$.

Ein umfangreicher Beweis wurde bereits in [PZZ11] gezeigt. Auch hier ist wieder die Idee, eine Registermaschine zu simulieren. Dabei wurden neben den üblichen Additions-, Subtraktions- und Halteinstruktionen auch Hilfsmodule konstruiert, die zwischen den einzelnen Komponenten sicherstellen sollen, dass die *Spikes*, unabhängig von der Zeit, gesteuert weitergeleitet werden.

Beispiel 4.2.2. Gegeben sei das folgende *zeitunabhängige* SN P-System Π_5 mit $\Pi_5 = (\{a\}, \sigma_1, \sigma_2, syn, in, out)$, das ein Vielfaches von k generieren soll:

$$\begin{aligned} \sigma_1 &= (0, \{a^k \to a^k, a^k \to \lambda\}) \\ \sigma_2 &= (k, \{a^k \to a^k\}) \\ syn &= \{(1,2), (2,1)\} \\ in &= 0 \\ out &= 2 \end{aligned}$$

Zum Zeitpunkt 0 (Zur Konfiguration C_0) besitzt nur das Neuron 2 eine Anzahl von k *Spikes*.

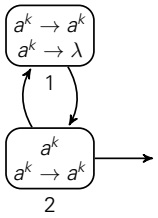


Abbildung 4.2.2: *zeitunabhängiges* SN P-System Π_5 nach [PZZ11]

Diese *Spikes* werden im nächsten Zeitschritt sowohl in die Umgebung, als auch zum Neuron 1 gesendet. Entscheidet sich Neuron 1 nichtdeterministisch die Regel $a^k \to a^k$ anzuwenden, so gelangt man im nächsten Zeitschritt zu einer Konfiguration C_i äquivalent zu Konfiguration C_0. Erst wenn sich Neuron 1 nichtdeterministisch für die Regel $a^k \to \lambda$ entscheidet, hält die Berechnung. Auffällig hierbei ist, dass nur nach jedem zweiten Zeitschritt *Spikes* in die Umgebung gesendet werden. Der Zwischenschritt ist notwendig, um die beiden Neuronen aufgrund der Variationen ihrer Verzögerungen zeitlich aufeinander abzustimmen. Nach $2i$ Zeitschritten hält die Konfiguration mit der folgenden Ergebnismenge:

$$N_{tot} = \{ki \mid i \in \mathbb{N}_1\} \in N_{tot}SN^{free}P_2(rule_2, cons_k, forg_k).$$

4.3 MODIFIKATION BEZÜGLICH DES VERHALTENS

SN P-Systeme können ebenfalls im Bezug auf ihr Verhalten modifiziert werden. Hierbei kann man zwischen dem zeitlichen Zusammenarbeiten der Neuronen und der Parallelität der Neuronen unterscheiden. Betrachtet werden also modifizierte SN P-Systeme, deren Neuronen zu unterschiedlichen Zeiten aktiv sind. Dabei wird ein Neuron als aktiv bezeichnet, wenn es eine Regel zum Anwenden wählt, wenn eine Regel zur Anwendung möglich ist oder aber keine Regel anwendet, wenn keine Regel zur Anwendung möglich ist.

4.3.1 Synchrone Spiking Neural P-Systeme

SN P-Systeme, deren Neuronen nach jedem spezifischen Zeitschritt aktiv sind, heißen synchron. Bei den bisherigen Beispielen in dieser Arbeit wurde immer Bezug auf *synchrone* SN P-Systeme genommen, da sie die einfachste Form von modifizierten SN P-Systemen im Bezug auf ihr Verhalten darstellen. Im Bezug auf *synchrone* SN P-Systeme erhält die Klasse der SN P-Systeme keinen Zusatz für den Modifikationsbezeichner (Mb_2). Standardmäßig wird also immer von einem *synchronen* SN P-System ausgegangen, sofern nicht anders angegeben.
Die entscheidenden Eigenschaften eines *synchronen* SN P-Systems sind hierbei:

- Jedes Neuron ist in jedem Zeitschritt aktiv. Wenn ein Neuron im Zeitpunkt t mindestens eine Regel anwenden kann, so muss es genau eine Regel anwenden.

- Die Verzögerungen d einer Regel aus den einzelnen Neuronen nehmen jeweils auf die globale Zeit t Bezug.

Mittels *synchroner* SN P-Systeme lassen sich Turing-berechenbare Funktionen berechnen und rekursiv aufzählbare Sprachen erzeugen. In den Kapiteln 4.1.1 und 4.1.2 wurden jeweils die Beweise für N_2SNP und $L_\lambda SN^eP$ erläutert.

4.3.2 Asynchrone Spiking Neural P-Systeme

Im Gegensatz zu den *synchronen* SN P-Systemen, sind die Neuronen in *nicht synchronen* SN P-Systemen nicht immer aktiv. Auch wenn es möglich ist, eine Regel in einem Neuron anzuwenden, so muss diese Regel nicht zwingend angewendet werden. In diesem Fall ist das Neuron, welches trotz der nötigen Voraussetzungen keine Regel anwendet, inaktiv. Jedoch sind *asynchrone* SN P-Systeme näher an einem biologischen Nervensystem, als *synchrone* SN P-Systeme, da die Neuronen in einem biologischen Nervensystem zu unterschiedlichen Zeiten aktiv sein können, sich also nicht an einer globalen Zeit t orientieren. Die Klasse dieser SN P-Systeme erhält den Zusatz für den Modifikationsbezeichner $Mb_2 = nsyn$.
Die besonderen Eigenschaften eines *asynchronen* SN P-Systems sind hierbei:

- In jedem Zeitschritt steht es den Neuronen frei, eine Regel anzuwenden oder nicht. So kann eine nicht angewendete Regel, die zu einem Zeitschritt anwendbar war, im nächsten Zeitschritt durch das Empfangen weiterer *Spikes* nicht mehr anwendbar sein.
- Die Ausführung einer Regel erfolgt in genau einem Zeitschritt. Demnach gibt es keine Verzögerungen d für die einzelnen Regeln. Daher muss für einen Zustand s_i aus einem Neuron σ_i nicht mehr die verbleibende Zeit $\tau(t)$ angegeben werden.
- Jedes Neuron ist in jedem Zeitschritt stets geöffnet.

Durch den Verlust der Eigenschaft, dass die Neuronen synchron zueinander arbeiten, verliert ein solches SN P-System an Berechnungsstärke. Die Frage, ob ein *asynchrones* SN P-System mit restriktiven Regeln Turing-berechenbare Funktionen berechnen kann, ist derzeitig noch offen. In [Cav+09] konnte jedoch das Ergebnis dieser Frage für *asynchrone erweiterte* SN P-Systeme gezeigt werden.

Theorem 4.3.1. $N_{tot}SNP^{e,nsyn}P = NRE$.

Gezeigt wurde dies mit einem allgemeinen SN P-System ohne einer *Forgetting Rule*. Die Idee ist auch hierbei, dass eine Registermaschine simuliert wird, in der die einzelnen Instruktionen (Addition, Subtraktion und Haltung) mit einem *asynchronen erweiterten* SN P-System simuliert werden.

Erwähnenswert ist hierbei, dass wie bei jedem anderen Neuron, auch dem Output-Neuron freigestellt ist, ob es in einem Zeitschritt eine Regel anwendet oder nicht. Es macht daher keinen Sinn, die Zeitschritte zwischen zwei in die Umgebung gesendeten *Spikes* zu berechnen. Aus diesem Grund wird die Gesamtanzahl der gesendeten *Spikes* berechnet. Das folgende Beispiel soll diesen Sachverhalt deutlich machen.

Beispiel 4.3.1. Gegeben sei das folgende *asynchrone* SN P-System Π_6, wobei $\Pi_6 = (\{a\}, \sigma_1, \sigma_2, syn, in, out)$ eine Menge von n natürlichen Zahlen generiert:

$$\begin{aligned} \sigma_1 &= (1, \{a \to a\}) \\ \sigma_2 &= (1, \{a \to a\}) \\ syn &= \{(1,2),(2,1)\} \\ in &= 0 \\ out &= 2 \end{aligned}$$

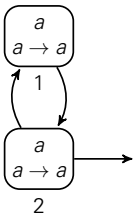


Abbildung 4.3.1: *asynchrones* SN P-System Π_6 nach [Cav+09]

Zum Zeitpunkt 0 (Konfiguration C_0) besitzen sowohl Neuron 1 als auch Neuron 2 einen *Spike*. In einem *asynchronen* SN P-System können diese beiden Neuronen nach beliebig vielen Zeitschritten eine Regel anwenden. Somit können beide Neuronen zum gleichen Zeitpunkt eine Regel anwenden, oder aber zu unterschiedlichen Zeitpunkten. Wenden beide Neuronen zur gleichen Zeit ihre Regel an, so wird jeweils ihr *Spike* konsumiert und zu den Neuronen 1, 2, sowie in die Umgebung ein *Spike* gesendet. Die Folgekonfiguration ist äquivalent zur Konfiguration C_0. Erst wenn die beiden Neuronen zu unterschiedlichen Zeitpunkten ihre Regel anwenden, sind zwei *Spikes* in genau einem Neuron, während sich im anderen Neuron keine *Spikes* mehr befinden. Es kann anschließend keine Regel mehr angewendet werden und das System hat eine Haltekonfiguration erreicht. Das System stoppt mit der erzeugten Ergebnismenge:

$$N_{tot}(\Pi_6) = \{0, 1, \ldots, n \mid n \in \mathbb{N}_0\} \in N_{tot}SN^{nsyn}P_2(rule_1, cons_1, forg_0)$$

4.3.3 Sequenzielle Spiking Neural P-Systeme

Die bisherigen Beispiele in dieser Arbeit zeigten SN P-Systeme, deren Neuronen parallel zueinander arbeiten. Diese parallele Arbeitsweise ist allerdings nicht biologisch natürlich, da die elektrischen Impulse der Nervenzellen in einem Bruchteil einer Sekunde weitergeleitet werden und somit die Neuronen zeitlich, wenn auch gering, zueinander versetzt Signale senden. SN P-Systeme, deren Neuronen zeitlich versetzt arbeiten, also dass in jedem Zeitschritt nur ein Neuron eine Regel anwenden kann, heißen *sequenzelle* SN P-Systeme. Die Klasse erhält hierbei als Zusatz den Modifikationsbezeichner $Mb_2 = seq$.
Man unterscheidet zwischen zwei verschiedenen Varianten von *sequenziellen* SN P-Systemen:

- Eine haltende Berechnung des *sequenziellen* SN P-Systems wird als valide anerkannt, wenn in jedem Zeitschritt der Berechnung mindestens eine Regel anwendbar ist. Das SN P-System heißt *stark sequenziell*.
- Wird eine haltende Berechnung des *sequenziellen* SN P-Systems als valide anerkannt, unabhängig ob es zu jedem Zeitschritt der Berechnung mindestens eine anwendbare Regel gibt, so heißt das SN P-System *schwach sequenziell*.

Diese beiden Eigenschaften können in die Aufzählung von Einschränkungen in die Klasse der SN P-Systeme aufgenommen werden. Dabei steht *weak* für schwach sequenziell und *strong* für stark sequenziell. Es gilt allgemein: *strong* ist stärker eingeschränkt als *weak*.

Theorem 4.3.2. $N_{tot}SN^{seq}P_*(strong) = NRE$.

Theorem 4.3.3. $N_{tot}SN^{seq}P_*(weak) = NRE$.

Der Beweis für Theorem 4.3.2 erfolgte bereits in [ZFP12] über die Konstruktion einer Registermaschine. Für Theorem 4.3.3 lässt sich einfach das SN P-System aus dem Beweis in [ZFP12] ableiten.
In [IPRP09] wurden *sequenzielle* SN P-Systeme im Bezug auf die minimale und maximale Anzahl der *Spikes* in einem Neuron untersucht. Wie bereits erwähnt, arbeiten die Neuronen in einem

biologischen Nervensystem nicht vollkommen parallel zueinander, sondern leicht versetzt. Des Weiteren ist bekannt, dass Neuronen, die ein höheres elektrisches Potenzial haben, d.h. mehr *Spikes* als andere Neuronen besitzen, schneller ein elektrisches Signal senden, als eben diese anderen Neuronen. *Sequenzielle* SN P-Systeme können unter weiteren Gesichtspunkten nach [IPRP09] betrachtet werden.

Definition 4.3.1 (*maximal sequenziell***).** Ein SN P-System arbeitet in einer *maximal sequenziellen* Weise, wenn in jedem Zeitschritt genau ein Neuron gewählt wird, das eine Regel anwenden kann. Dieses Neuron muss zu diesem Zeitpunkt die höchste Anzahl an *Spikes* besitzen. Kommen mehrere Neuronen in Frage, so wird nichtdeterministisch ein Neuron gewählt.

Definition 4.3.2 (*pseudomaximal sequenziell***).** Ein SN P-System arbeitet in einer *pseudomaximal sequenziellen* Weise, wenn in jedem Zeitschritt alle Neuronen eine Regel anwenden, die die höchste Anzahl an *Spikes* zu diesem Zeitpunkt besitzen.

Die Definitionen für *minimal sequenziell* und *pseudominimal sequenziell* können intuitiv angegeben werden. Doch vom besonderen Interesse sollen *maximal sequenzielle* und *pseudomaximal sequenzielle* SN P-Systeme sein. Diese Variationen können ebenfalls in die Aufzählung als Einschränkung in den Klassenbezeichner der SN P-Systeme aufgenommen werden. Dabei steht *max* für *maximal sequenziell*, *pmax* für *pseudomaximal sequenziell*, *min* für *minimalsequenziell* und *pmin* für *pseudominimal sequenziell*.

Theorem 4.3.4. $N_2SN^{seq}P_*(strong, max) = NRE$.

Theorem 4.3.5. $N_2SN^{seq}P_*(strong, pmax) = NRE$.

In [IPRP09] wurde bereits Theorem 4.3.4 und Theorem 4.3.5 jeweils über die Simulation einer Registermaschine bewiesen. Hierbei wurden für beide Variationen die Hauptinstruktionen (Addition, Subtraktion und Halten) als SN P-Systeme konstruiert.

Zum besseren Verständnis soll das folgende Beispiel dienen.

Beispiel 4.3.2. Gegeben sei das folgende *maximal stark sequenzielle* SN P-System Π_7, wobei $\Pi_7 = (\{a\}, \sigma_1, \sigma_2, syn, in, out)$ eine Menge von n natürlichen Zahlen generiert:

$$\begin{aligned} \sigma_1 &= (1, \{a^2/a \to a\}) \\ \sigma_2 &= (3, \{a^3/a \to a, a^2 \to \lambda\} \\ syn &= \{(1,2), (2,1)\} \\ in &= 0 \\ out &= 2 \end{aligned}$$

Zum Zeitpunkt 0 besitzt das Neuron 2 insgesamt drei *Spikes* und das Neuron 1 einen *Spike*. Nur

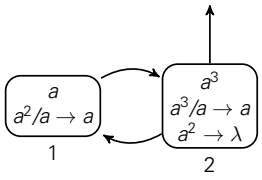


Abbildung 4.3.2: *sequenzielles* SN P-System Π_7

das Neuron 2 kann eine Regel anwenden. Nach Anwendung der Regel $a^3/a \to a$ wird ein *Spike* in die Umgebung und ein *Spike* zum Neuron 1 gesendet. im nächsten Zeitpunkt (Konfiguration C_1) besitzen nun beide Neuronen jeweils zwei *Spikes*. Entscheidet sich das System die Regel $a^2/a \to a$ aus Neuron 1 anzuwenden, so wird ein *Spike* konsumiert und ein *Spike* zum Neuron 2 gesendet. Die nun erreichte Konfiguration ist äquivalent zur Anfangskonfiguration. Erst wenn sich das System nach $2n+1$ Zeitschritten (in Konfiguration C_{2n+1}) für die Regel $a^2 \to \lambda$ aus Neuron 2 entscheidet, hält die Berechnung nach einem weiteren Zeitschritt. In der Haltekonfiguration besitzen sowohl Neuron 1 als auch Neuron 2 nur noch einen *Spike* und können somit keine Regel mehr anwenden. In die Umgebung wurden n viele *Spikes* gesendet:

$$N_{tot}(\Pi_7) = \{1, \ldots, n \mid n \in \mathbb{N}_1\} \in N_{tot}SN^{seq}P_2(strong, max, rule_2, cons_2, forg_2)$$

5 ZUSAMMENFASSUNG

In dieser Arbeit wurde vor allem die Berechnungsstärke von einzelnen Modifikationen der SN P-Systeme im Bezug auf die Ergebnismengen der einzelnen SN P-Systeme untersucht. Untersucht man die Ergebnismenge eines SN P-Systems, so muss diese Ergebnismenge immer unter einer Interpretation (N_2, L_{bin}, ...) betrachtet werden. Im Bezug auf diese Interpretation kann man die Berechnungsstärke von SN P-Systemen unter zwei Gesichtspunkten betrachten. So überprüft man zum einen welche Sprachen und zum anderen welche Mengen natürlicher Zahlen mit einem SN P-System generiert werden können. Aufgrund der Eigenschaft, dass ein SN P-System nach Definition 3.1.1 nicht in der Lage ist, ein Neuron zu besitzen, welches sowohl eine *Firing Rule* $r_1 = E/v \rightarrow w; d$ als auch eine *Forgetting Rule* $r_2 = u \rightarrow \lambda$ mit $u \in L(E)$ enthält, ist zwar die Klasse der *restriktiven* SN P-Systeme unter der Interpretation N_2 (N_2SNP) äquivalent zu *NRE*, jedoch sind *restriktive* SN P-Systeme unter keiner Interpretation (L_{bin}, L_{res}, L_λ) in der Lage, Sprachen aus *RE* zu erzeugen. *Restriktive* SN P-Systeme beschränken sich bei der Berechnungsstärke im Bezug auf Sprachen nur auf eine Teilmenge von endlichen Sprachen *FIN* unter der Interpretation L_{bin}. Bisher ist nur für *erweiterte* SN P-Systeme bewiesen, dass sie in der Lage sind, alle Sprachen aus *RE* unter der Interpretation L_λ (unter der Interpretation L_{res} sogar nur reguläre Sprachen) zu generieren. Die Klasse $L_\lambda SN^eP$ ist demnach äquivalent zu *RE*. Jedoch sind SN P-Systeme mit restriktiven Regeln näher an einem biologischen Nervensystem, als *erweiterte* SN P-Systeme, da in einer biologischen Nervenzelle nur ein Signal entlang des Axons gesendet und nicht maßgeblich zwischen der Stärke dieses elektrischen Signals unterschieden wird. Eine bestimmte Anzahl an *Spikes* auf einmal über ein Axon - im Bezug auf *erweiterte* SN P-Systeme - zu senden, erscheint daher nicht natürlich. Eine mögliche Lösung und ein noch zu untersuchender Sachverhalt ist daher die Betrachtung eines SN P-Systems mit mehreren Output-Neuronen, so dass der gesendete *Spiketrain* als eine Sequenz von Vektoren betrachtet wird. Ausgehend von der Berechnungsstärke von *restriktiven* und *erweiterten* SN P-Systemen, lassen sich genau diese Berechnungsstärken bei Kombinationen weiterer Modifikationen beobachten. So sind beispielsweise *restriktiv homogene*, *asynchrone*, *sequenzielle* oder *zeitbefreite* SN P-Systeme ebenfalls in der Lage, alle Mengen natürlicher Zahlen aus *NRE* zu generieren.
Doch auch die Betrachtung weiterer biologischer Komponenten führt zu weiteren Modifikationen und Untersuchungen der SN P-Systeme. So spielen Astrozyten eine wichtige Rolle bei der Flüssigkeitsregulation im Gehirn und wirken sich dabei hemmend oder erregend auf die Nervenzellen aus. Die Modellierung eines SN P-Systems mit Astrozyt-ähnlichen Funktionen wurde in [Pŏ7a] durchgeführt. Eine genaue Untersuchung der Berechnungsstärke und funktionalen Anwendung von SN P-Systemen mit Astrozyten wurde [PWH12] und [MRPJ12] durchgeführt. Ein weiteres Problem stellen hemmende *Spikes* dar. In der Neurobiologie werden sowohl erregende, als auch hemmende Impulse berücksichtigt. Doch mehr als nur einen Typ von *Spikes* zu verwenden, widerspricht dem Prinzip von SN P-Systemen gegenüber den P-Systemen, nur ein einelementiges Alphabet zu nutzen. Die Verwendung von hemmenden Impulsen wurde unter der Verwendung von *Anti-Spikes* in [PP09] und speziell in *homogenen* SN P-Systemen in [SW14] untersucht. Bei der Betrachtung von *erweiterten* SN P-Systemen stellt

sich zu dem noch das Problem, dass Synapsen einen Impuls eines beliebig hohen elektrischen Potenzials senden können, jedoch die Anzahl der maximal zu sendenden *Spikes* in einem realen Nervensystem begrenzt ist. Dieses Verhalten wird bei der Modifikation der Synapsen berücksichtigt. Diese SN P-Systeme mit den soeben genannten Grenzen, werden auch als SN P-Systeme mit gewichteten Synapsen bezeichnet. SN P-Systeme mit gewichteten Synapsen wurden in [Wan+10] und in [Pan+12] im Bezug auf ihrer Berechnungsstärke erforscht. Von besonderem Interesse ist auch die Betrachtung eines biologischen Nervensystems, das sich mittels Zellteilung weiterentwickelt. So kann ausgehend von einem SN P-System im Laufe einer Berechnung mittels Knospung und Zellteilung die Anzahl der Neuronen und Regeln dynamisch verändert werden, um beispielsweise NP-schwere Probleme lösen zu können. SN P-Systeme mit Zellteilung und Knospung wurden daher schon in [PPPJ11] untersucht. Die Betrachtung von SN P-Systemen ist ein relativ junges Forschungsthema, und somit sind viele Vertiefungen in verschiedene Modifikationen von SN P-Systemen wissenschaftlich möglich.

6 LITERATURVERZEICHNIS

[Baa13] Franz Baader. *Formale Systeme. Teil 1 - Automatentheorie und formale Sprachen*. Skript. Technische Universität Dresden, Fakultät Informatik, Sept. 2013.

[Baa14] Franz Baader. *Theoretische Informatik und Logik. Teil 1 - Berechenbarkeit, Teil 2 - Komplexitätstheorie*. Skript. Technische Universität Dresden, Fakultät Informatik, Apr. 2014.

[Cav+09] Matteo Cavaliere et al. "Asynchronous spiking neural P systems". In: *Theoretical Computer Science* 410.24-25 (2009), pp. 2352–2364.

[Che+07] Haiming Chen et al. "On String Languages Generated by Spiking Neural P Systems". In: *Fundamenta Informaticae* 75 (Jan. 2007), pp. 141–162.

[Che+08] Haiming Chen et al. "Spiking neural P systems with extended rules: universality and languages". In: *Natural Computing* 7.2 (June 2008), pp. 147–166.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Ed. by Michael A. Harrison. 79-67950. Addison-Wesley Publishing Company, Inc., Mar. 1979. ISBN: 0-201-02988-X.

[IPRP09] Oscar H. Ibarra, Andrei Păun, and Rodríguez-Patón. "Sequential SNP systems based on min/max spike number". In: *Theoretical Computer Science* 410 (Aug. 2009), pp. 2982–2991.

[IPY06] Mihai Ionescu, Gheorghe Păun, and Takashi Yokomori. "Spiking Neural P Systems". In: *Fundam. Inf.* 71 (Feb. 2006), pp. 279–308.

[Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1967. ISBN: 0-13-165563-9.

[MRK14] Venkata Padmavati Metta, Srinivasan Raghuraman, and Kamala Krithivasan. "Spiking Neural P Systems with Cooperating Rules". In: *Membrane Computing*. Vol. 8961. Springer International Publishing, Aug. 2014, pp. 314–329.

[MRPJ12] Luis F. Macías-Ramos and Mario J. Pérez-Jiménez. "Spiking Neural P Systems with Functional Astrocytes". In: *Membrane Computing*. Vol. 7762. Springer Berlin Heidelberg, 2012, pp. 228–242.

[PĂ00] Gheorghe Păun. "Computing with Membranes". In: *Journal of Computer and System Sciences* 61 (2000), pp. 108–143.

[PĂ02] Gheorghe Păun. *Membrane Computing. An Introduction*. Natural Computing Series. Springer-Verlag Berlin Heidelberg, 2002. ISBN: 978-3-642-56196-2.

[PĂ07a] Gheorghe Păun. "Spiking Neural P Systems with Astrocyte-Like Control". In: *Journal of Universal Computer Science* 13.11 (Nov. 2007), pp. 1707–1721.

[PĂ07b] Gheorghe Păun. "Twenty Six Research Topics About Spiking Neural P Systems". In: *Fifth brainstorming week on membrane computing*. Sevilla (Spanien), 2007, pp. 263–280.

[Pan+12] LinQiang Pan et al. "Spiking Neural P Systems with Weighted Synapses". In: *Neural Processing Letters* 35.1 (2012), pp. 13–27.

[Pea94] Giuseppe Peano. *Notations de logique mathématique*. fre. Turin: Guadagnini, 1894. URL: http://eudml.org/doc/204440.

[PP07] Andrei Păun and Gheorghe Păun. "Small universal spiking neural P systems". In: *Biosystems* 90.1 (2007), pp. 48–60.

[PP09] LinQiang Pan and Gheorghe Păun. "Spiking Neural P Systems with Anti-Spikes". In: *Journal of Computers, Communication & Control* 4.3 (2009), pp. 273–282.

[PPJR06] Gheorghe Păun, Mario J. Pérez-Jiménez, and Grzegorz Rozenberg. "Spike Train in Spiking Neural P Systems". In: *Intern. J. Found. Computer Sci* 17 (2006), pp. 975–1002.

[PPPJ11] LinQiang Pan, Gheorghe Păun, and Mario J. Pérez-Jiménez. "Spiking neural P systems with neuron division and budding". In: *Science China Information Sciences* 54.8 (Aug. 2011), pp. 1596–1607.

[PWH12] LinQiang Pan, Jun Wang, and Hendrik Jan Hoogeboom. "Spiking Neural P Systems with Astrocytes". In: *Neural Computation* 24.3 (Mar. 2012), pp. 805–825.

[PZZ11] LinQiang Pan, Xiangxiang Zeng, and Xingyi Zhang. "Time-Free Spiking Neural P Systems". In: *Neural Computation* 23.5 (May 2011), pp. 1320–1342.

[Rog87] Hartley Jr. Rogers. *Theory of Recursive Functions and Effective Computability*. 3rd ed. Cambridge, Massachusetts, London, England: MIT Press, 1987. ISBN: 0-262-68052-1.

[Sch11] Daniel Schröder. "Expanding Spiking Neural P Systems". Diplomarbeit. Technische Universität Dresden, Fakultät Informatik, Feb. 2011.

[SW14] Tao Song and Xun Wang. "Homogenous Spiking Neural P Systems with Inhibitory Synapses". In: *Neural Processing Letters* (2014).

[Tho68] Ken Thompson. "Regular Expression Search Algorithm". In: *Communications of the ACM* 11.6 (June 1968), pp. 419–422.

[Wan+10] Jun Wang et al. "Spiking Neural P Systems with Weights". In: *Neural Computation* 22.10 (Sept. 2010), pp. 2615–2646.

[ZFP12] Xingyi Zhang, Xianyong Fang, and LinQiang Pan. "Sequential spiking neural P systems with exhaustive use of rules". In: *BioSystems* 108 (Apr. 2012), pp. 52–62.

[ZZP09] Xiangxiang Zeng, Xingyi Zhang, and LinQiang Pan. "Homogenous Spiking Neural P Systems". In: *Fundamenta Informaticae* 97.1-2 (Jan. 2009), pp. 275–294.