Zur Mächtigkeit von Netzwerken evolutionärer Prozessoren mit zwei Knotenarten

Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik
Universitätsplatz 2, D-39106 Magdeburg

und

Bianca Truthe*
Universitat Rovira i Virgili, Facultat de Lletres, GRLMC
Plaça Imperial Tàrraco 1, E-43005 Tarragona

Zusammenfassung

Gegenstand dieser Arbeit ist die Erzeugungskraft von Netzwerken evolutionärer Prozessoren, bei denen die Anzahl der vorkommenden Knotenarten auf zwei beschränkt ist. Wir weisen nach, dass (bis auf Durchschnitt mit einem Monoid) jede rekursiv aufzählbare Sprache von einem Netzwerk mit einem löschenden und zwei einfügenden Knoten erzeugbar ist, Netzwerke mit beliebig vielen einfügenden und ersetzenden Prozessoren kontextabhängige Sprachen erzeugen und zum Erzeugen jeder kontextabhängigen Sprache (bis auf Durchschnitt mit einem Monoid) nur ein einfügender und ein ersetzender Knoten nötig sind, sowie Netzwerke mit beliebig vielen ersetzenden und löschenden Prozessoren endliche Sprachen erzeugen und für jede endliche Sprache ein ersetzender oder löschender Knoten ausreichend ist.

1 Einleitung

Netzwerke von Sprachprozessoren wurden von E. CSUHAJ-VARJÚ und A. SALOMAA eingeführt ([4]). Solch ein Netzwerk kann als Graph angesehen werden, bei dem jeder Knoten Regeln und Wörter hat, die er entsprechend den Regeln ableitet, und die nach dem Passieren gewisser Filter über die Kanten zu anderen Knoten gelangen. Die von einem Netzwerk erzeugte Sprache besteht aus allen Wörtern, die irgendwann in einem festgelegten Knoten auftreten.

Von biologischen Prozessen inspiriert, haben J. CASTELLANOS, C. MARTIN-VIDE, V. MITRANA und J. SEMPERE in [2] Netzwerke evolutionärer Prozessoren eingeführt. Dabei sind die verwendeten Regeln Ersetzungen eines Buchstabens durch einen anderen, Einfügen eines Buchstabens und Löschen eines Buchstabens. Diese Regeln modellieren Punktmutationen in der Biologie.

In [3] wurde gezeigt, dass Netzwerke evolutionärer Prozessoren rekursiv aufzählbare Sprachen erzeugen und für jede dieser Sprachen sechs Knoten genügen. Dieses Ergebnis wurde in [1] zu drei Knoten verbessert, wobei alle drei Typen vorkommen. In der vorliegenden Arbeit wird die Mächtigkeit von Netzwerken untersucht, in denen nur zwei Arten von Knoten vorkommen.

Wir geben im Folgenden einige der in dieser Arbeit verwendeten Begriffe und Notationen an. Für weitere Definitionen sei auf die Literatur verwiesen (z. B. [8]).

*Die Forschung wurde unterstützt durch die Alexander-von-Humboldt-Stiftung.

Zu einem Alphabet V bezeichnen wir mit V^* die Menge aller Wörter über V einschließlich dem Leerwort λ.

Eine Grammatik ist ein Quadrupel $G = (N, T, P, S)$ mit einem Alphabet N von Nichtterminalen, einem Alphabet T von Terminalen, einer endlichen und nicht-leeren Menge P von Ersetzungsregeln der Form $\alpha \to \beta$ mit $\alpha \in (N \cup T)^* \setminus T^*$ und $\beta \in (N \cup T)^*$ und einem Startsymbol $S \in N$.

Eine Grammatik ist in Kuroda-Normalform, wenn alle ihre Ersetzungsregeln eine der folgenden Formen haben: $AB \to CD$, $A \to CD$, $A \to x$, $A \to \lambda$ mit $A, B, C, D \in N$ und $x \in N \cup T$.

Eine konditionale (monotone) Grammatik ist eine Grammatik, bei der die Regeln Paare von einer „herkömmlichen“ Ersetzungsregel und einer regulären Menge sind. Eine konditionale Regel (p, R) ist nur dann auf ein Wort w anwendbar, wenn das Wort w zur regulären Menge R gehört und die Ersetzungsregel p auf w anwendbar ist.

Eine Regel $\alpha \to \beta$ heißt
- ersetzend, wenn $|\alpha| = |\beta| = 1$ gilt, und
- löschend, wenn $|\alpha| = 1$ und $\beta = \lambda$ gelten.

Wir betrachten Einfügen als Gegenstück zu Löschen, schreiben $\lambda \to a$ für einen Buchstaben a und bezeichnen es ebenfalls als Regel. Das Einfügen $\lambda \to a$ liefert zu einem Wort w ein Wort $w_1 a w_2$ mit $w = w_1 w_2$ für zwei (möglicherweise leere) Wörter w_1 und w_2. Ersetzende, löschende und einfügende Regeln werden auch Evolutionsregeln genannt.

Wir definieren nun Netzwerke evolutionärer Prozessoren.

Definition 1.1

(i) *Ein Netzwerk evolutionärer Prozessoren der Größe n ist ein $(n+3)$-Tupel*

$$\mathcal{N} = (V, N_1, N_2, \ldots, N_n, E, j)$$

mit

- *einem Alphabet V,*
- *n Knoten $N_i = (M_i, A_i, I_i, O_i)$ (für $1 \le i \le n$), wobei*
  - *M_i eine sortenreine Menge von Evolutionsregeln ist,*
  - *A_i eine endliche Teilmenge von V^* ist und*
  - *I_i und O_i reguläre Sprachen über V sind,*
- *einer Teilmenge E von $\{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ und*
- *einer natürlichen Zahl j mit $1 \le j \le n$.*

(ii) *Eine Konfiguration eines Netzwerkes $\mathcal{N}$ der Größe n ist ein n-Tupel $C = (C(1), C(2), \ldots, C(n))$ mit $C(i) \subseteq V^*$ für $1 \le i \le n$.*

(iii) *Es seien $C = (C(1), C(2), \ldots, C(n))$ und $C' = (C'(1), C'(2), \ldots, C'(n))$ zwei Konfigurationen eines Netzwerkes $\mathcal{N}$. Wir sagen, dass C zu C' in einem*

- *Evolutionsschritt abgeleitet wird (geschrieben $C \Longrightarrow C'$), wenn für $1 \le i \le n$ die Menge $C'(i)$ aus allen Wörtern $w \in C(i)$, auf die keine Regel aus M_i anwendbar ist, und aus allen Wörtern w, zu denen ein Wort $v \in C(i)$ und eine Regel $p \in M_i$ so existieren, dass $v \Longrightarrow_p w$ gilt, besteht,*
- *Kommunikationsschritt abgeleitet wird (geschrieben $C \vdash C'$), wenn für $1 \le i \le n$*

$$C'(i) = (C(i) \setminus O_i) \cup \bigcup_{(k,i) \in E} C(k) \cap O(k) \cap I(i)$$

gilt.

Die Berechnung von $\mathcal{N}$ ist eine Folge von Konfigurationen $C_t = (C_t(1), C_t(2), \ldots, C_t(n))$, $t \geq 0$, so dass

- *$C_0 = (A_1, A_2, \ldots, A_n)$ gilt,*
- *für alle $t \geq 0$ die Konfiguration C_{2t} zu C_{2t+1} in einem Evolutionsschritt führt: $C_{2t} \Longrightarrow C_{2t+1}$,*
- *für alle $t \geq 0$ die Konfiguration C_{2t+1} zu C_{2t+2} in einem Kommunikationsschritt führt: $C_{2t+1} \vdash C_{2t+2}$.*

(iv) *Die von einem Netzwerk $\mathcal{N}$ erzeugte Sprache $L(\mathcal{N})$ ist*

$$L(\mathcal{N}) = \bigcup_{t \geq 0} C_t(j),$$

wobei $C_t = (C_t(1), C_t(2), \ldots, C_t(n))$, $t \geq 0$ die Berechnung von $\mathcal{N}$ ist.

Man stelle sich ein Netzwerk evolutionärer Prozessoren als einen gerichteten Graphen vor, dessen Knoten N_i ($1 \leq i \leq n$) Prozessoren sind, die über die Kanten (angegeben durch die Menge E) Wörter austauschen. Jeder Prozessor N_i hat eine Menge M_i von Evolutionsregeln, eine Menge von Wörtern (anfänglich A_i), einen Eingangsfilter I_i und einen Ausgangsfilter O_i. Der Prozessor heißt

- ersetzend, wenn $M_i \subseteq \{a \to b \mid a, b \in V\}$ gilt,
- einfügend, wenn $M_i \subseteq \{\lambda \to b \mid b \in V\}$ gilt, und
- löschend, wenn $M_i \subseteq \{a \to \lambda \mid a \in V\}$ gilt.

In einem Evolutionsschritt leitet jeder Prozessor seine Wortmenge entsprechend seiner Regeln ab. Die neue Wortmenge besteht dabei aus allen jenen Wörtern, die dadurch entstehen, dass eine Regel auf ein Wort der ursprünglichen Menge an einer möglichen Stelle angewendet wird, und jenen Wörten, auf die keine Regel anwendbar ist.

In einem Kommunikationsschritt sendet jeder Prozessor N_i alle Wörter, die seinen Ausgangsfilter passieren, zu allen benachbarten Prozessoren (zu denen eine Kante hinführt). Die Wörter, die der Ausgangsfilter nicht durchlässt, bleiben im Prozessor. Außerdem nimmt jeder Prozessor alle Wörter auf, die ihn auf einer ankommenden Kante erreichen und seinen Eingangsfilter passieren. Wörter, die einen Knoten verlassen und von keinem Knoten aufgenommen werden, gehen verloren (verschwinden aus dem Netzwerk).

Die Arbeit eines Netzwerkes beginnt mit einem Evolutionsschritt; danach wechseln sich Kommunikations- und Evolutionsschritte ab. Die erzeugte Sprache besteht aus allen Wörtern, die irgendwann zu dem ausgewiesenen Prozessor N_j gehören.

2 Netzwerke mit ersetzenden Prozessoren

In diesem Abschnitt wird die Erzeugungskraft von Netzwerken betrachtet, die nur aus ersetzenden und löschenden (aber keinen einfügenden) oder nur aus ersetzenden und einfügenden (aber keinen löschenden) Knoten bestehen.

Satz 2.1 *Die Klasse der von Netzwerken evolutionärer Prozessoren mit ausschließlich ersetzenden und löschenden Knoten erzeugten Sprachen stimmt mit der Klasse $\mathcal{L}(FIN)$ der endlichen Sprachen überein.*

Beweis. Ohne einfügende Regeln entstehen aus den (endlich vielen) Wörtern der Anfangsmengen keine längeren Wörter. Folglich gibt es zu jedem Netzwerk $\mathcal{N}$ eine natürliche Zahl $k_{\mathcal{N}}$, so dass jedes Wort der Sprache $L(\mathcal{N})$ von der Länge höchstens $k_{\mathcal{N}}$ ist.

Jede endliche Sprache F ist erzeugbar von einem Netzwerk mit genau einem löschenden oder ersetzenden Knoten, in dem F die Anfangswortmenge des Knotens ist und die Regeln so gewählt werden,

dass sie nicht anwendbar sind (z. B. durch Hinzufügen von Buchstaben zum Alphabet des Netzwerkes, die nicht in Wörtern der Sprache F auftreten). □

Satz 2.2 *Die Klasse der von Netzwerken evolutionärer Prozessoren mit ausschließlich ersetzenden und einfügenden Knoten erzeugten Sprachen stimmt mit der Klasse $\mathcal{L}(CS)$ der kontextabhängigen Sprachen überein.*

Beweis. Die Arbeitsweise eines Netzwerks evolutionärer Prozessoren kann durch eine konditionale monotone Grammatik simuliert werden. Die Konstruktion einer solchen Grammatik zu einem Netzwerk ist ausführlich in der Arbeit [6] erläutert. Da konditionale monotone Grammatiken kontextabhängige Sprachen erzeugen ([5]), erzeugt auch jedes Netzwerk mit ersetzenden und einfügenden aber ohne löschende Prozessoren eine kontextabhängige Sprache.

Zu jeder kontextabhängigen Sprache L existieren eine Menge T und ein Netzwerk $\mathcal{N}$ mit genau einem einfügenden und einem ersetzenden Prozessor, so dass $L = L(\mathcal{N}) \cap T^*$ gilt. In der Arbeit [7] (und [6]) ist dargelegt, wie zu einer Grammatik G in Kuroda-Normalform, die die Sprache L erzeugt, ein Netzwerk konstruiert werden kann, das die Ableitungsschritte der Grammatik G simuliert. Der ersetzende Prozessor markiert in geeigneter Weise die Nichtterminale, auf die eine Regel der Grammatik angewendet werden soll (so, dass an der Markierung stets die Regel ablesbar ist) und ersetzt schließlich die Markierung durch neue Nichtterminale (entsprechend der betreffenden Regel); der einfügende Prozessor sorgt für das Verlängern von Wörtern (bei Regeln der Form $A \to BC$). Die Filter bewirken, dass nur jene Wörter mit richtig gesetzten Markierungen erhalten bleiben (die anderen verlassen einen Knoten und verschwinden). Wird als Menge T die Menge der Terminalsymbole der Grammatik G genommen, so bilden die vom Netzwerk erzeugten Wörter, die nur aus Terminalsymbolen bestehen, gerade die Sprache L.

Nimmt man einen weiteren Prozessor hinzu, der die terminalen Wörter „auffängt“, so erhält man ein Netzwerk $\mathcal{N}'$, das genau die Sprache L erzeugt. Dieser Prozessor darf nur Regeln haben, die auf kein Wort der Sprache L anwendbar sind, damit er die ankommenden Wörter nicht noch verändern kann. Da einfügende Regeln immer anwendbar sind, muss ein ersetzender Prozessor genommen werden. □

3 Netzwerke ohne ersetzende Prozessoren

In diesem Abschnitt wird die Erzeugungskraft von Netzwerken betrachtet, die nur aus löschenden und einfügenden (aber keinen ersetzenden) Knoten bestehen.

Satz 3.1 *Die Klasse der von Netzwerken evolutionärer Prozessoren mit ausschließlich löschenden und einfügenden Knoten erzeugten Sprachen stimmt mit der Klasse $\mathcal{L}(RE)$ der rekursiv aufzählbaren Sprachen überein.*

Beweis. Beliebige Netzwerke evolutionärer Prozessoren erzeugen rekursiv aufzählbare Sprachen, folglich auch Netzwerke ohne ersetzende Knoten ([3]).

Zu jeder rekursiv aufzählbaren Sprache L existieren eine Menge T und ein Netzwerk $\mathcal{N}$ mit genau einem löschenden und zwei einfügenden Prozessoren, so dass $L = L(\mathcal{N}) \cap T^*$ gilt. In der Arbeit [7] ist dargelegt, wie zu einer Grammatik G in Kuroda-Normalform, die die Sprache L erzeugt, ein Netzwerk konstruiert werden kann, das die Ableitungsschritte der Grammatik G simuliert. Ein Prozessor fügt Symbole ein und markiert damit die Stelle, an der eine Regel der Grammatik angewendet werden soll. Der löschende Prozessor entfernt die zu überschreibenden Nichtterminale. Der zweite einfügende Prozessor fügt die Nichtterminale oder das Terminal der rechten Seite der betreffenden Regel ein. Der löschende

Prozessor entfernt daraufhin die Markierungssymbole. In [6] ist die Arbeitsweise des Netzwerkes detaillierter erläutert.
Wird als Menge T die Menge der Terminalsymbole der Grammatik G genommen, so bilden die vom Netzwerk erzeugten Wörter, die nur aus Terminalsymbolen bestehen, gerade die Sprache L.

Nimmt man einen weiteren Prozessor hinzu, der die terminalen Wörter „auffängt“, so erhält man ein Netzwerk $\mathcal{N}'$, das genau die Sprache L erzeugt. Dieser Prozessor darf nur Regeln haben, die auf kein Wort der Sprache L anwendbar sind, damit er die ankommenden Wörter nicht noch verändern kann. Da einfügende Regeln immer anwendbar sind, muss ein löschender Prozessor genommen werden. □

Folgerung 3.2 *Es gibt ein Netzwerk $\mathcal{N}$ evolutionärer Prozessoren mit zwei einfügenden und einem löschenden Knoten, so dass $L(\mathcal{N})$ eine nicht-rekursive Sprache ist.*

Beweis. Da die Familie der rekursiven Sprachen unter Durchschnitt mit Mengen T^*, wobei T ein Alphabet ist, abgeschlossen ist, erzeugt das Netzwerk im Beweis von Satz 3.1 zu einer nicht-rekursiven Sprache L eine ebenfalls nicht-rekursive Sprache. □

Literatur

[1] A. ALHAZOV, C. MARTIN-VIDE und YU. ROGOZHIN, On the number of nodes in universal networks of evolutionary processors. *Acta Inf.* **43** (2006) 331–339.

[2] J. CASTELLANOS, C. MARTIN-VIDE, V. MITRANA und J. SEMPERE, Solving NP-complete problems with networks of evolutionary processors. In: *Proc. IWANN*, Lecture Notes in Computer Science **2084**, Springer-Verlag, Berlin, 2001, 621–628.

[3] J. CASTELLANOS, C. MARTIN-VIDE, V. MITRANA und J. SEMPERE, Networks of evolutionary processors. *Acta Informatica* **39** (2003) 517–529.

[4] E. CSUHAJ-VARJÚ und A. SALOMAA, Networks of parallel language processors. In: *New Trends in formal Language Theory* (Hrsg. GH. PĂUN und A. SALOMAA), Lecture Notes in Computer Science **1218**, Springer-Verlag, Berlin, 1997, 299–318.

[5] J. DASSOW und GH. PĂUN, *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[6] J. DASSOW und B. TRUTHE, On the Power of Networks of Evolutionary Processors. Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, Technical report, 2007.

[7] J. DASSOW und B. TRUTHE, On the Power of Networks of Evolutionary Processors. In: *Machines, Computations and Universality, MCU 2007, Orléans, France, September 10–13, 2007, Proceedings*. 2007.

[8] G. ROZENBERG und A. SALOMAA, *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.