Zu akzeptierenden Netzwerken evolutionärer Prozessoren mit zwei Knotenarten

Bianca Truthe

Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
truthe@iws.cs.uni-magdeburg.de

Zusammenfassung

In dieser Arbeit wird untersucht, welche Sprachen von Netzwerken evolutionärer Prozessoren, bei denen die Anzahl der vorkommenden Knotenarten auf zwei beschränkt ist, akzeptiert werden.

Wir zeigen, dass jede kontextabhängige Sprache von einem Netzwerk mit einem ersetzenden Prozessor und einem Ausgabeknoten akzeptiert wird. Jede rekursiv aufzählbare Sprache wird von einem Netzwerk mit drei evolutionären Prozessoren (einem ersetzenden, einem einfügenden und einem Ausgabe-Prozessor) akzeptiert. Auch Netzwerke mit einfügenden und löschenden Prozessoren aber ohne ersetzende Prozessoren akzeptieren alle rekursiv aufzählbaren Sprachen. Dabei sind zwei einfügende, ein löschender und ein Ausgabe-Prozessor ausreichend.

1. Einleitung

Netzwerke von Sprachprozessoren wurden von E. CSUHAJ-VARJÚ und A. SALOMAA eingeführt ([3]). 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 Punktmutationen in der Biologie inspiriert, haben J. CASTELLANOS, C. MARTIN-VIDE, V. MITRANA und J. SEMPERE in [2] Netzwerke evolutionärer Prozessoren eingeführt. Dabei sind die verwendeten Regeln Ersetzen eines Buchstabens durch einen anderen, Einfügen eines Buchstabens und Löschen eines Buchstabens. In [6] wurde eine Charakterisierung der Komplexitätsklasse $\mathbb{NP}$

basierend auf akzeptierenden Netzwerken evolutionärer Prozessoren präsentiert. Von J. DASSOW und V. MITRANA wurden in [4] erstmals akzeptierende Netzwerke evolutionärer Prozessoren untersucht, bei denen die Kommunikation der Prozessoren durch reguläre Filter gesteuert wird.

In [5] wurde die Erzeugungskraft von Netzwerken evolutionärer Prozessoren untersucht, bei denen nur zwei Knoten-Arten vorkommen. Insbesondere wurden folgende Resultate bewiesen:

- Netzwerke ohne löschende Knoten erzeugen alle kontextabhängigen Sprachen; jeweils ein ersetzender und ein einfügender Knoten sind ausreichend.
- Netzwerke ohne ersetzende Knoten erzeugen alle rekursivaufzählbaren Sprachen; jeweils zwei einfügende und ein löschender Knoten sind ausreichend.

In der vorliegenden Arbeit werden die dualen Aussagen bewiesen:

- Jede kontextabhängige Sprache wird von einem Netzwerk evolutionärer Prozessoren mit einem ersetzenden, einem löschenden und einem Ausgabe-Knoten akzeptiert.
- Jede rekursiv aufzählbare Sprache wird von einem Netzwerk evolutionärer Prozessoren mit zwei einfügenden, einem löschenden und einem Ausgabe-Knoten akzeptiert.

Netzwerke, die nur aus ersetzenden Prozessoren bestehen, können ausschließlich endliche Sprachen erzeugen, aber sie können unendliche Sprachen akzeptieren. Der Grund dafür liegt darin, dass erzeugende Netzwerke mit einer endlichen Wortmenge beginnen und ersetzende Prozessoren die Wortlänge nicht vergrößern können, wogegen akzeptierende Netzwerke unendlich viele Eingabewörter erhalten.

Wir zeigen, dass zum Akzeptieren einer jeden kontextabhängigen Sprache ein ersetzender Prozessor und ein Ausgabeknoten genügen. Außerdem weisen wir nach, dass jede rekursiv aufzählbare Sprache von einem Netzwerk mit einem ersetzenden, einem einfügenden und einem Ausgabe-Knoten akzeptiert wird.

Wir geben im Folgenden einige der in dieser Arbeit verwendeten Begriffe und Notationen an. Für weitere Definitionen sei auf die Literatur verwiesen (z. B. [7]).

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 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 akzeptierende Netzwerke evolutionärer Prozessoren.

Definition 1.1

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

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

mit

- *zwei endlichen Mengen U (dem Eingabe-Alphabet) und V (dem Arbeitsalphabet, $U \subseteq V$),*
- *n Knoten $N_i = (M_i, I_i, O_i)$ (für $1 \leq i \leq n$), wobei*
  - *M_i eine sortenreine Menge von Evolutionsregeln ist sowie*
  - *I_i und O_i reguläre Sprachen über V sind,*
- *einer Teilmenge E von $\{1,2,\ldots,n\} \times \{1,2,\ldots,n\}$,*
- *einer natürlichen Zahl j mit $1 \leq j \leq n$ und*
- *einer nicht-leeren Teilmenge O von $\{1,2,\ldots,n\}$.*

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

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

- *Evolutionsschritt abgeleitet wird (geschrieben $C \Longrightarrow C'$), wenn für $1 \leq i \leq n$ 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 $1 \leq i \leq n$*

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

gilt.

Die Berechnung eines Netzwerkes $\mathcal{N}^{(n)}$ auf einem Eingabewort $w \in U^*$ ist eine Folge von Konfigurationen $C_t^w = (C_t^w(1), C_t^w(2), \ldots, C_t^w(n))$, $t \geq 0$, derart, dass

- *$C_0^w = (C_0^w(1), C_0^w(2), \ldots, C_0^w(n))$ mit $C_0^w(j) = \{w\}$ und $C_0^w(i) = \emptyset$ für $1 \leq i \neq j \leq n$ gilt,*
- *für alle $t \geq 0$ die Konfiguration C_{2t}^w zu C_{2t+1}^w in einem Evolutionsschritt führt: $C_{2t}^w \Longrightarrow C_{2t+1}^w$,*
- *für alle $t \geq 0$ die Konfiguration C_{2t+1}^w zu C_{2t+2}^w in einem Kommunikationsschritt führt: $C_{2t+1}^w \vdash C_{2t+2}^w$.*

(iv) *Die von einem Netzwerk $\mathcal{N}^{(n)}$ schwach akzeptierte Sprache $L_{\mathrm{w}}(\mathcal{N})$ und stark akzeptierte Sprache $L_{\mathrm{s}}(\mathcal{N})$ sind als*

$$L_{\mathrm{w}}(\mathcal{N}^{(n)}) = \{\, w \in U^* \mid \exists t \geq 0 \, \exists o \in O : C_t^w(o) \neq \emptyset \,\} \text{ und}$$
$$L_{\mathrm{s}}(\mathcal{N}^{(n)}) = \{\, w \in U^* \mid \exists t \geq 0 \, \forall o \in O : C_t^w(o) \neq \emptyset \,\}$$

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

Mit einem Knoten N_i und einer Zeit $t \geq 0$ verbinden wir eine Wortmenge $C_t(i)$. Zu Beginn enthält der ausgewiesene Knoten N_j das Eingabewort; alle anderen Knoten enthalten keine Wörter.

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. Wenn zu einem Zeitpunkt ein Ausgabeknoten ein Wort enthält, so wird das Eingabewort schwach akzeptiert. Wenn zu einem Zeitpunkt alle Ausgabeknoten ein Wort enthalten, so wird das Eingabewort stark akzeptiert. Die schwach oder stark akzeptierte Sprache eines Netzwerkes ist die Menge aller Wörter, die schwach bzw. stark akzeptiert werden.

2. Netzwerke ohne einfügende Prozessoren

In [5] wurde gezeigt, dass Netzwerke ohne löschende Prozessoren die kontextabhängigen Sprachen erzeugen. In diesem Abschnitt betrachten wir den dualen Fall: akzeptierende Netzwerke ohne einfügende Prozessoren. In [4] wurde gezeigt, dass jede von einem solchen Netzwerk akzeptierte Sprache kontextabhängig ist und zu jeder kontextabhängigen Sprache ein solches Netzwerk existiert, das die Sprache akzeptiert. Die Anzahl der Prozessoren ist bei dem konstruierten Netzwerk linear in der Anzahl der Regeln der zugrunde liegenden Grammatik.

In diesem Abschnitt wird gezeigt, dass die Anzahl der Prozessoren nicht von der Grammatik abhängt.

Satz 2.1 *Zu jeder kontextabhängigen Sprache L gibt es ein Netzwerk $\mathcal{N}$ evolutionärer Prozessoren mit jeweils genau einem ersetzenden, einem löschenden und einem Ausgabe-Knoten ohne Regeln, das schwach und stark die Sprache L akzeptiert: $L = L_{\mathrm{w}}(\mathcal{N}) = L_{\mathrm{s}}(\mathcal{N})$.*

Beweis. Zu jeder kontextabhängigen Sprache L gibt es eine monotone Grammatik $G = (N, T, P, S)$ in Kuroda-Normalform, die L erzeugt. Ähnlich zum Beweis in [5], dass zum Erzeugen ein ersetzender und ein einfügender Knoten ausreichen, kann ein akzeptierendes Netzwerk zu L konstruiert werden. Dabei werden die Regeln der zugehörigen Grammatik G rückwärts simuliert. In dem ersetzenden Knoten werden die Regeln mit längengleicher rechter und linker Seite simuliert ($A \to x$ für $A \in N$ und $x \in N \cup T$ sowie $AB \to CD$ für $A, B, C, D \in N$). Der Ausgangsfilter sorgt dafür, dass die Wörter in einem erfolgreichen Ableitungsprozess den Knoten nicht verlassen. Alle anderen Wörter verlassen den Knoten, passieren aber nicht die Eingangsfilter der anderen Knoten und verschwinden somit aus dem Netzwerk. Zum Rückwärtssimulieren von Regeln der Form $A \to BC$ (für $A, B, C \in N$) werden der ersetzende und der löschende Knoten gebraucht.

Ein Wort w gehört genau dann zur Sprache L, wenn es eine „Rückwärtsableitung“ zum Axiom S gibt. Dieses Wort ist das einzige, das den Eingangsfilter des Ausgabe-Knotens passiert. Somit erhält der Ausgabe-Knoten genau dann ein Wort, wenn das Eingabewort zur Sprache gehört. Das Netzwerk akzeptiert also die Sprache L. Da es nur einen Ausgabe-Knoten gibt, stimmen schwache und starke Akzeptanz überein. □

Wenn wir nur löschende Prozessoren erlauben, können nicht alle kontextabhängigen Sprachen akzeptiert werden, auch nicht alle linearen Sprachen. Zum Beispiel kann die lineare Sprache $\{ a^n b a^n \mid n \geq 1 \}$ nicht akzeptiert werden. Zu einem Eingabewort $a^n b a^m$ mit $n \geq 1$, $m \geq 1$ kann ein Netzwerk nur durch Löschen und die regulären Filter nicht entscheiden, ob $n = m$ gilt (wenn ein a gelöscht wird, sieht das Netzwerk nicht, auf welcher Seite vom b es war).

Folglich ist das beschriebene Netzwerk bezüglich der Anzahl der ersetzenden Knoten optimal. Aber wir können zeigen, dass löschende Prozessoren nicht nötig sind.

Satz 2.2 *Zu jeder kontextabhängigen Sprache L gibt es ein Netzwerk $\mathcal{S}$ evolutionärer Prozessoren mit jeweils genau einem ersetzenden Knoten und einem Ausgabe-Knoten ohne Regeln, das schwach und stark die Sprache L akzeptiert: $L = L_{\mathrm{w}}(\mathcal{S}) = L_{\mathrm{s}}(\mathcal{S})$.*

Beweis. Hierzu wird das Netzwerk aus dem vorhergehenden Beweis so modifiziert, dass der ersetzende Knoten die Aufgabe des löschenden Knotens übernimmt. Allerdings wird ein Zeichen nicht wirklich gelöscht, sondern durch eine Löschmarkierung $\textvisiblespace$ ersetzt. Die Filter müssen dann so beschaffen sein, dass alle Löschmarkierungen in einem Wort ignoriert werden. Der Ausgabe-Knoten lässt alle Wörter zu, die das Axiom S enthalten und ansonsten nur aus Löschmarkierungen bestehen. □

Diese Anzahl von Prozessoren ist optimal, da Eingabe- und Ausgabe-Knoten verschieden sein müssen (sonst würde jedes Eingabewort akzeptiert werden).

3. Netzwerke mit einfügenden Prozessoren

Der wesentliche Unterschied zwischen kontextabhängigen und nicht kontextabhängigen Grammatiken in Kuroda-Normalform besteht darin, dass in der Normalform einer beliebigen Regelgrammatik löschende Regeln (λ-Regeln) erlaubt sind. Um eine λ-Regel rückwärts zu simulieren, verwenden wir einen einfügenden Knoten.

Satz 3.1 *Zu jeder rekursiv aufzählbaren Sprache L gibt es ein Netzwerk $\mathcal{N}$ evolutionärer Prozessoren mit jeweils genau einem ersetzenden, einem einfügenden und einem Ausgabe-Knoten ohne Regeln, das schwach und stark die Sprache L akzeptiert: $L = L_{\mathrm{w}}(\mathcal{N}) = L_{\mathrm{s}}(\mathcal{N})$.*

Beweis. Die Idee ist, das Netzwerk $\mathcal{S}$ aus dem Beweis zu Satz 2.2 um einen einfügenden Prozessor zu erweitern, der die Rückwärtssimulation der λ-Regeln übernimmt.

Zwischen den Rückwärtssimulationen zweier Regeln kann der ersetzende Knoten ein Symbol markieren, damit das Wort den Knoten verlassen und zum einfügenden Knoten wechseln darf. Der Prozessor fügt dann ein Nichtterminal ein, das zu einer λ-Regel der zugrunde liegenden Grammatik G gehört, und sendet das Wort an den ersetzenden Knoten zurück. Dieser Prozessor muss zunächst die Markierung aufheben. Falls eine Markierung nicht im richtigen Moment gesetzt oder aufgehoben wird, geht das Wort verloren. Auch hier wird ein Eingabewort genau dann akzeptiert, wenn es auf das Startwort S der Grammatik zurückgeführt werden kann (als gelöscht markierte Stellen werden ignoriert). □

In [1] wurde gezeigt, dass jede rekursiv aufzählbare Sprache von einem Netzwerk mit einem einfügenden und einem löschenden Prozessor erzeugt werden kann. Analog dazu kann die folgende Aussage bewiesen werden.

Satz 3.2 *Zu jeder rekursiv aufzählbaren Sprache L gibt es ein Netzwerk $\mathcal{N}$ evolutionärer Prozessoren mit zwei einfügenden, einem löschenden und einem Ausgabe-Knoten, das schwach und stark die Sprache L akzeptiert: $L = L_{\mathrm{w}}(\mathcal{N}) = L_{\mathrm{s}}(\mathcal{N})$.*

Beweis. Es sei $w = a_1 a_2 \cdots a_n$ das Eingabewort. Einer der beiden einfügenden Knoten wandelt zunächst das Wort zu $a_1 \text{␣} a_2 \text{␣} \cdots a_n \text{␣}$ um. Die anderen beiden Knoten simulieren dann die Ableitung rückwärts analog zum Verfahren aus [1]. □

Würde das Eingabewort bereits in der Weise geliefert werden, dass die Buchstaben durch einzelne Leerzeichen getrennt sind, wären ein einfügender, ein löschener und ein Ausgabe-Knoten ausreichend.

Literatur

[1] A. ALHAZOV, J. DASSOW, C. MARTIN-VIDE, YU. ROGOZHIN und B. TRUTHE, On Networks of Evolutionary Processors with Nodes of Two Types. Eingereicht.

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

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

[4] J. DASSOW und V. MITRANA, Accepting networks of non-inserting evolutionary processors. In: I. PETRE und G. ROZENBERG (Hrsg.), *Proceedings of NCGT 2008 – Workshop on Natural Computing and Graph Transformations, Leicester, United Kingdom, September 8, 2008*. University of Leicester, 2008, 29–41.

[5] J. DASSOW und B. TRUTHE, On the Power of Networks of Evolutionary Processors. In: J. DURAND-LOSE und M. MARGENSTERN (Hrsg.), *Machines, Computations and Universality, MCU 2007, Orléans, France, September 10–13, 2007, Proc*. LNCS 4664, Springer, 2007, 158–169.

[6] M. MARGENSTERN, V. MITRANA und M. J. PÉREZ-JIMÉNEZ, Accepting Hybrid Networks of Evolutionary Processors. In: C. FERRETTI, G. MAURI und C. ZANDRON (Hrsg.), *10th International Workshop on DNA Computing*. LNCS 3384, Springer, 2005, 235–246.

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