Akzeptierende programmierte Grammatiken ohne Nichtterminale

Henning Bordihn

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
Postfach 4120
D-39016 Magdeburg
email: bordihn@irb.cs.uni-magdeburg.de

Henning Fernau

Wilhelm-Schickhard-Institut
für Informatik,
Universität Tübingen,
Sand 13,
72076 Tübingen,
email: fernau@informatik.uni-tuebingen.de

In unseren Arbeiten über akzeptierende Grammatiken und Systeme [1, 2, 5] haben wir nachgewiesen, daß die triviale Äquivalenz zwischen generierenden und akzeptierenden Typ-i-Grammatiken der Chomsky-Hierarchie (s. z.B. [6]) auf eine Vielzahl von Grammatiktypen nicht übertragen werden kann. Ein Beispiel für dieses Phänomen stellen die programmierten Grammatiken (mit Vorkommenstest und kontextfreien Kernregeln) dar, die im akzeptierenden Modus eine echt größere Beschreibungsmächtigkeit besitzen als im generierenden Modus. Diese Erkenntnis wird in den genannten Arbeiten aus dem Beweis gewonnen, daß die akzeptierenden programmierten Grammatiken ohne löschende Regeln exakt die Familie der kontextsensitiven Sprachen definieren bzw. jede rekursiv-aufzählbare Sprache beschreiben können, falls λ-Regeln erlaubt sind, während bekanntermaßen die Sprachfamilien der zugehörigen generierenden Grammatiken echte Teilmengen der Familien $\mathcal{L}(\mathrm{CS})$ bzw. $\mathcal{L}(\mathrm{RE})$ sind (vgl. z.B. [4]). Die Simulation der Typ-1- bzw. Typ-0-Grammatiken durch akzeptierende programmierte Grammatiken erfordert die Erweiterung des Alphabets der Nichtterminale um eine Reihe von Hilfssymbolen. In diesem Vortrag untersuchen wir nun die reinen Versionen dieser Grammatiken, bei denen (wie im Falle der Lindenmayer-Systeme) nicht zwischen terminalen und nichtterminalen Symbolen unterschieden wird.

Eine reine programmierte Grammatik [3, 4] ist ein Tripel $G = (V, P, S)$. Hier ist V ein Alphabet, $S \subseteq V^+$ eine endliche Menge von Axiomen und P eine endliche Menge von Produktionen der Form $(r : u \to v,\ \sigma(r),\ \phi(r))$, wobei r die Marke der Produktion, $u \to v$ eine Ersetzungsregel mit $u \in V^*$, $v \in V^*$ ist und $\sigma(r)$ und $\phi(r)$ Mengen von Marken ausgewählter Produktionen aus P sind. Eine Folge $y_0, y_1, \ldots, y_n$ von Wörtern über V heißt genau dann Ableitung in G, wenn für jedes $i \in \{1, \ldots, n\}$ eine Produktion $(r_i : u_i \to v_i,\ \sigma(r_i),\ \phi(r_i))$ existiert, so daß

$$y_{i-1} = z_{i-1} u_i z'_{i-1},\ y_i = z_{i-1} v_i z'_{i-1} \text{ und, falls } 1 \le i < n,\ r_{i+1} \in \sigma(r_i)$$

oder

$$u_i \text{ ist kein Teilwort von } y_{i-1},\ y_{i-1} = y_i, \text{ und, falls } 1 \le i < n,\ r_{i+1} \in \phi(r_i)$$

gelten. Dann schreiben wir $y_0 \stackrel{n}{\Rightarrow} y_n$ und definieren die von G generierte Sprache als

$$L_{\text{gen}}(G) = \{w \in V^* \mid s \stackrel{n}{\Rightarrow} w \text{ für ein } s \in S \text{ und ein } n \ge 0\}$$

und die von G akzeptierte Sprache als

$$L_{\text{acc}}(G) = \{w \in V^* \mid w \stackrel{n}{\Rightarrow} s \text{ für ein } s \in S \text{ und ein } n \ge 0\}\,.$$

$L_{\text{gen}}(G) \in \mathcal{L}_{\text{gen}}(\text{pP,CF,ac})$ gelte genau dann, wenn jede Kernregel $u \to v$ aus der Regelmenge von G eine reine kontextfreie Regel ist, d.h. falls u ein Symbol ist. Entsprechend sei $L_{\text{acc}}(G) \in \mathcal{L}_{\text{acc}}(\text{pP,CF,ac})$ genau dann, wenn für alle Kernregeln $u \to v$ aus G gilt, daß v ein Symbol ist. Gibt es keinen Vorkommenstest (appearance checking) in G, d.h. gilt $\phi(r) = \emptyset$ für jede Marke r von Produktionen in G, so schreiben wir genauer $L_{\text{gen}}(G) \in \mathcal{L}_{\text{gen}}(\text{pP,CF})$ bzw. $L_{\text{acc}}(G) \in \mathcal{L}_{\text{acc}}(\text{pP,CF})$. Gilt für jede Marke r, daß $\sigma(r) = \phi(r)$, so schreiben wir statt dessen $L_{\text{gen}}(G) \in \mathcal{L}_{\text{gen}}(\text{pP,CF,ut})$ bzw. $L_{\text{acc}}(G) \in \mathcal{L}_{\text{acc}}(\text{pP,CF,ut})$ (unconditional transfer). Sind λ-Produktionen verboten, ersetzen wir CF durch CF–λ in dieser Notation.

Betrachtet man reine programmierte Grammatiken ohne Vorkommenstest, so kann man die Beweistechnik zum Nachweis trivialer Äquivalenzen zwischen akzeptierendem und generierndem Modus analog zum nichtreinen Fall anwenden (s. [1, 2]). Obwohl die Beweise für die Grammatiken mit Vorkommenstest unter Verwendung von Hilfssymbolen nicht auf den reinen Fall übertragbar sind, gelingt es, die generierenden reinen programmierten Grammatiken auch in diesem Fall durch akzeptierende zu simulieren. Damit sind die entsprechenden Inklusionen im klassischen Fall noch einmal direkt (ohne den „Umweg“ über die Typ-1- bzw. Typ-0-Sprachen) nachgewiesen. Die Echtheit dieser Teilmengenbeziehungen für die reinen Versionen wird mit Hilfe der Sprache

$$L = \{a^n b^n \mid n \ge 1\} \cup \{a^{m+1} b a^n b^k \mid m \ge 1,\ n \ge 1,\ m+n = k\}\,,$$

für die sogar $L \in \mathcal{L}_{\mathrm{acc}}(\mathrm{pP,CF}-\lambda, \mathrm{ut}) \setminus \mathcal{L}_{\mathrm{gen}}(\mathrm{pP, CF, ac})$ gilt, nachgewiesen.

Allerdings benötigt die verwendete Simulationsmethode die Möglichkeit von „echten“ Vorkommenstests, so daß sie für reine Grammatiken mit unconditional transfer nicht verwendbar ist. In diesem Fall erhalten wir ein Unvergleichbarkeitsresultat, obwohl bei Zulassung von Nichtterminalen ebenfalls eine echte Inklusion zwischen der Familie der generierten Sprachen und der Familie der akzeptierten Sprachen besteht.

Wir fassen die Ergebnisse wie folgt zusammen: Für $\alpha \in \{\mathrm{CF,CF}-\lambda\}$ gilt:

- $\mathcal{L}_{\mathrm{gen}}(\mathrm{pP},\alpha) = \mathcal{L}_{\mathrm{acc}}(\mathrm{pP},\alpha)$
- $\mathcal{L}_{\mathrm{gen}}(\mathrm{pP},\alpha,\mathrm{ac}) \subset \mathcal{L}_{\mathrm{acc}}(\mathrm{pP},\alpha,\mathrm{ac})$
- $\mathcal{L}_{\mathrm{gen}}(\mathrm{pP},\alpha,\mathrm{ut})$ und $\mathcal{L}_{\mathrm{acc}}(\mathrm{pP},\alpha,\mathrm{ut})$ sind unvergleichbar.

Damit haben wir zum ersten Mal einen Grammatiktyp gefunden, für den Unvergleichbarkeit zwischen den Familien der akzeptierten und der generierten Sprachen nachzuweisen war. Weitere Beispiele lassen sich bei der Untersuchung deterministischer L-Systeme ohne Nichtterminale oder ohne Tafeln finden. Ein umfassendes Papier zu diesem Thema ist in Vorbereitung.

Literatur

[1] H. Bordihn and H. Fernau. Accepting grammars and systems. Technical Report 9/94, Universität Karlsruhe, Fakultät für Informatik, 1994.

[2] H. Bordihn and H. Fernau. Accepting grammars with regulation. *International Journal of Computer Mathematics*, 53:1–18, 1994.

[3] J. Dassow. Pure grammars with regulated rewriting. *Revue Roumaine Math. Pures Et Appl.*, 31:657–666, 1986.

[4] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*, volume 18 of *EATCS Monographs in Theoretical Computer Science*. Berlin: Springer, 1989.

[5] H. Fernau and H. Bordihn. Remarks on accepting parallel systems. *International Journal of Computer Mathematics*, 56:51–67, 1995.

[6] A. K. Salomaa. *Formal Languages*. Academic Press, 1973.