Formale Sprachen

(parallele und regulierte Ersetzung)

SoSe 2007 in Trier

Henning Fernau
Universität Trier
fernau@informatik.uni-trier.de

Formale Sprachen Gesamtübersicht

- Organisatorisches / Einführung
  Motivation / Erinnerung / Fragestellungen

- Diskussion verschiedener Sprachklassen:
  gesteuerte Ersetzungsverfahren
  parallele Ersetzungsverfahren

- Algebraische Ansätze:
  abstrakte Sprachfamilien
  formale Potenzreihen

Motivation

Zweck formaler Sprachen: Modellierung (Analyse, Beschreibung) syntaktischer Phänomene aus:

1. Natürlichen Sprachen (Linguistik)

2. Programmiersprachen

3. Biologie (und andere Bereiche)

Beobachtung: Die Welt ist nicht kontextfrei.

Dabei wichtig:

1. Einfachheit der Modelle

2. Natürlichkeit der Modelle

3. Algorithmische Fragestellungen

Linguistische Motivation

John, Mary, David, ...
are
a widower, a widow, a widower, ...
respectively.

Übersetzen wir weibliche Bezeichnungen nach a und männliche nach b sowie "are" und "respectively" nach c, so erhalten wir als Formalisierung solcher Bezeichnungen:

$$\mathrm{L} = \{ww \mid w \in \{\mathrm{a}, \mathrm{b}\}^{+}\{\mathrm{c}\}\}$$

Ist Englisch kontextfrei ?

Trügerisches Argument: $\mathbb{L}$ aus vorigem Beispiel ist nicht kontextfrei.

Wenn Englisch als Sprache $\mathbb{E}$ kontextfrei wäre, so aber auch $\mathbb{E} \cap \mathbb{R}$ für jede reguläre Sprache $\mathbb{R}$.

Mit geeignetem $\mathbb{R}$ könnten wir $\mathbb{L}$ quasi “herausschneiden” aus $\mathbb{E}$.

Daher ist $\mathbb{E}$ nicht kontextfrei.

Hinweis: Abschlusseigenschaften kann man oft gebrauchen, um Echtheit von Inklusionen nachzuweisen.

Sind Programmiersprachen kontextfrei ?

Betrachte folgendes Fragment:

```
{
    int x;
    y=1
}
```

Unter der Maßgabe, dass eine Variable deklariert sein muss vor ihrer Benutzung, ist obiges Programmstück nur dann richtig, wenn beide vorkommenden Variablen identisch sind.

Wieso sind also Programmiersprachen (mit dieser Eigenschaft) im Allgemeinen nicht kontextfrei ?

Ein logisches Beispiel

Betrachten den Aussagenkalkül mit den logischen Verknüpfungen $\wedge$, $\vee$, $\Longrightarrow$, $\neg$ (sowie Klammern).
Atomare Aussagen werden durch Wörter über $\{a\}$ unär kodiert.
τ sei die Sprache der Theoreme in dem skizzierten Kalkül. Bekannt ist: $\tau \in \mathcal{L}_1$.

Betrachte die reguläre Sprache $R = \{(x \vee y) \Longrightarrow z \mid x, y, z \in \{a\}^*\}$.
$\tau \cap R = \{(x \vee x) \Longrightarrow x \mid x \in \{a\}^*\} \notin \mathcal{L}_2$, daher $\tau \notin \mathcal{L}_2$.

Unbekannt: Ist τ, eingeschränkt auf endliche Menge atomarer Aussagen, kontextfrei ?

Relationen und Funktionen als Sprachen

Die Sprache $\{a_1^n a_2^n \mid n \geq 0\}$ könnte man auch als Unärkodierung der Diagonalen auf den natürlichen Zahlen $\mathbb{N}$ deuten.

Ist allgemeiner R eine k-stellige Relation auf den natürlichen Sprachen, so entspricht dieser Relation die Sprache

$$L_R = \{a_1^{m_1} a_2^{m_2} \cdots a_k^{m_k} \mid (m_1, \ldots, m_k) \in R\}$$

über dem Alphabet $\{a_1, \ldots, a_k\}$.

Interpretieren wir eine k-stellige Funktion $f : \mathbb{N}^k \to \mathbb{N}$ als $(k+1)$-stellige Relation, so liefert f eine Sprache L_f.

Viele scheinbar einfache Relationen und Funktionen liefern nicht kontextfreie Sprachen.
Beispiel: $f(x) = x^2$, $g(x, y) = xy$, aber: $h(x, y) = x + y$. Was ist mit: $\phi(x, y) = \max(x, y)$?

D0L Systeme

hatten wir beiläufig in der allerersten Vorlesung kennengelernt.

Ist $\mathfrak{h} : \Sigma^* \to \Sigma^*$ ein Morphismus und $\omega \in \Sigma^*$ ein Axiom, so ist die vom D0L System $G = (\Sigma, \mathfrak{h}, \omega)$ beschriebene Sprache:

$$L(G) = \{\mathfrak{h}^i(\omega) \mid i \in \mathbb{N}\} = \{\omega, \mathfrak{h}(\omega, \mathfrak{h}(\mathfrak{h}(\omega)), \dots\}.$$

Da $\mathfrak{h}$ Abbildung, ist auch $f_G : \mathbb{N} \to \mathbb{N}, n \mapsto \ell(\mathfrak{h}^n(\omega))$ eine Abbildung, die *Wachstumsfunktion* von G.

So können z.B. die Fibonacci-Zahlen beschrieben werden, aber auch die Zweierpotenzen (wie ?) .

Siehe auch 2. Hauptthema der Vorlesung: Die Welt ist nicht sequentiell.

Flussdiagramme für (kontextfreie) Grammatiken
führt auf *programmierte* Grammatiken.

- Arbeit auf Satzformen als Zustandsraum
- Operationen: (sequentielles) Anwenden von Regeln
- Abfrage (zur Verzweigung): Anwendbarkeit von Regeln
- Beispiele (Tafel): $L_1 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$, $L_2 = \{a^{2^n} \mid n \in \mathbb{N}\}$.
  Wo liegen die Sprachen in der Chomsky-Hierarchie ?

Im Folgenden teilweise (bewusst) englische Folien (MFCS 1996).

Unified Framework

A *grammar controlled by a bicoloured digraph* (*graphgesteuerte Grammatik*) [2, 8, 9, 11, 16, 17] or *G grammar* is an 8-tuple $G = (V_N, V_T, P, S, \Gamma, \Sigma, \Phi, \mathfrak{h})$ where

- V_N, V_T, P, S define the set of nonterminals, terminals, context-free core rules, and the start symbol;
- Γ is a bicoloured digraph, i.e., $\Gamma = (U, E)$, where U is a finite set of nodes and $E \subseteq U \times \{g, r\} \times U$ is a set of arcs coloured by g or r ("green" or "red");
- $\Sigma \subseteq U$ are the initial nodes;
- $\Phi \subseteq U$ are the final nodes;
- $\mathfrak{h} : U \to (2^P \setminus \{\emptyset\})$ relates nodes with rule sets.

Two different definitions of the *appearance checking mode* (*Vorkommenstest*): $(x, u) \Rightarrow (y, v)$ ($(x, u) \Rightarrow_c (y, v)$, respectively) holds in G with $(x, u), (y, v) \in (V_N \cup V_T)^* \times U$, if either, for some $(u, g, v) \in E$,

$$x = x_1 \alpha x_2, \quad y = x_1 \beta x_2, \quad \alpha \to \beta \in h(u),$$

or every (one, respectively) rule of $h(u)$ is not applicable to x, $y = x$, $(u, r, v) \in E$. The reflexive transitive closure of $\Rightarrow$ ($\Rightarrow_c$, respectively) is denoted by $\overset{*}{\Rightarrow}$ ($\overset{*}{\Rightarrow}_c$, respectively). The corresponding languages generated by G are defined by $L_{[c]}(G) =$

$$\{x \in V_T^* \mid \exists u \in \Sigma \exists v \in \Phi(S, u) \overset{*}{\Rightarrow}_{[c]} (x, v)\}.$$

Language families:

- $\mathcal{L}(\mathrm{G}_{[\mathrm{c}]}, \mathrm{CF,ac})$: general case. (*mit Vorkommenstest*)
- $\mathcal{L}(\mathrm{G}_{[\mathrm{c}]}, \mathrm{CF})$: all arcs are green. (*ohne Vorkommenstest*)
- $\mathcal{L}(\mathrm{G}_{[\mathrm{c}]}, \mathrm{CF,ut})$: all arcs are both green and red. (*mit unbedingtem Übergang*)

We write CF$-\lambda$ instead of CF if we do not allow erasing core rules.

Hinweis: Auch reguläre, kontextsensitive oder uneingeschränkte Kernregeln denkbar! Wie sehen die entsprechenden Sprachfamilien aus ?

Special cases of G grammars:

- **P***rogrammed grammars* [2, 10, 14, 15]:
  - Every node contains exactly one rule.
  - Every node is both initial and final.

  Usual notation of rules: $(\mathrm{r} : A \to w, \sigma, \phi)$.

  Language families: $\mathcal{L}(\mathrm{P}, \mathrm{CF}, \mathrm{ac})$, $\mathcal{L}(\mathrm{P}, \mathrm{CF}, \mathrm{ut})$, and $\mathcal{L}(\mathrm{P}, \mathrm{CF})$.
- Grammars with **r***egular* **c***ontrol* [5, 2, 14]:
  - Every node contains exactly one rule.
  - If there is a red arc from node u to node v, then there is also a green arc from node u to node v.
- **M***atrix grammars* [1, 2, 14]:
  - Only the initial nodes are allowed to have more than one in-going green arc.
  - Only final nodes are allowed to have more than one out-going green arc.
  - Between every final node and every initial node, there is a green arc.

- **T***ime-***v***ariant grammars* [2, 12, 13, 14]:
  - If there is a red arc from node $\mathfrak{u}$ to node $\mathfrak{v}$, then there is also a green arc from node $\mathfrak{u}$ to node $\mathfrak{v}$. (The "green graph" is a ring.)
  - There is one initial node, and every node is final.

  Usual notation: a periodic function ϕ mapping natural numbers to rule sets such that $\phi(\mathfrak{n})$ prescribes the $\mathfrak{n}^{\text{th}}$ derivation step.

  Now, we can consider both the $\Rightarrow$ and the $\Rightarrow_{\mathfrak{c}}$ derivations, leading to the families of languages $\mathcal{L}(\text{TV}_{[\mathfrak{c}]}, \text{CF,ac})$, $\mathcal{L}(\text{TV}_{[\mathfrak{c}]}, \text{CF,ut})$, and $\mathcal{L}(\text{TV}_{[\mathfrak{c}]}, \text{CF})$.

Similar: **M**atrix **s**et and **r**egular **s**et **c**ontrol grammars.

Known facts

Theorem 1. If $X \in \{G_c, P, M, rC, TV_c\}$, $Y \in \{CF, CF - \lambda\}$, $Z \in \{ac, \lambda\}$, then $\mathcal{L}(X, Y, Z) = \mathcal{L}(G, Y, Z)$.

Theorem 2.

- $\mathcal{L}(G, CF) \subsetneq \mathcal{L}(G, CF, ac) = \mathcal{L}_0$.
- $\mathcal{L}(G, CF - \lambda) \subsetneq \mathcal{L}(G, CF - \lambda, ac) \subsetneq \mathcal{L}_1$.

(Sources: [2, 3, 4, 6, 7])

Programmed grammars

Original interpretation of rule $(p : A \to w, \sigma, \phi)$:

- in general:
  `if` $A \to w$ is applicable `then`
  `begin` apply $A \to w$;
  continue with a rule from σ
  `end`
  `else begin` continue with a rule from ϕ
  `end`.
- $\phi = \emptyset$:
  `if` $A \to w$ is applicable `then`
  `begin` apply $A \to w$;
  continue with a rule from σ
  `end`.
- $\phi = \sigma$:
  `if` $A \to w$ is applicable `then`
  `begin` apply $A \to w$
  `end`;
  continue with a rule from σ.

Main Lemma

Let $L \in \mathcal{L}_0$, $L \subseteq V_T^*$. Then:
there is a (P,CF,ut) grammar $\tilde{G}$ with:
$w \in L$ iff $w\$\# \in L(\tilde{G})$,
where $\{\$, \#\} \cap V_T = \emptyset$.

Simulation idea of given (P,CF,ac) grammar $G = (V_N, V_T, P, S)$:
(1) one initialization production
$(\text{init} : \tilde{S} \to S\$\sigma, \{p_i^+, p_i^- \mid A_i = S\})$,
(2) termination rules $(t_i : B_i \to F, \{t_{i+1}\})$ for $1 \leq i \leq n$ and
$(t_{n+1} : \sigma \to \#, \{t_{n+1}\})$, and
(3), for every p_i, $1 \leq i \leq m$, simulating rules

$$\begin{array}{lcll} (p_i^- & : & A_i \to F, \{\, q^+, q^- \mid q \in \phi_i \,\}) & , \\ (p_i^+ & : & A_i \to w_i\sigma, \{p_i'\}) & , \text{ and} \\ (p_i' & : & \sigma \to \lambda, \{q^+, q^- \mid q \in \sigma_i\} \cup \{t_1\}) & . \end{array}$$

□

Corollaries

- $\mathcal{L}$(P,CF,ut) contains non-recursive languages.
- $\mathcal{L}$(P,CF$-\lambda$,ut) $\subsetneq \mathcal{L}$(P,CF,ut).
- $\mathcal{L}$(P,CF,ut) $= \mathcal{L}_0$ iff
  $\mathcal{L}$(P,CF,ut) is closed under right derivatives.
  (anyhow, non-effective closure!)

Remark: Emptiness is decidable for (P,CF,ut) grammars [10].

The non-erasing case

Let $\mathrm{L} \in \mathcal{L}(\mathrm{P,CF{-}\lambda,ac})$, $\mathrm{L} \subseteq \mathrm{V}_\mathrm{T}^*$.
Then, there is a constant $\mathrm{m} > 0$ and a (P,CF$-\lambda$,ut) grammar $\tilde{\mathrm{G}}$ such that
$w \in \mathrm{L}$ iff $w\$\#^\mathrm{m} \in \mathrm{L}(\tilde{\mathrm{G}})$,
where $\{\$, \#\} \cap \mathrm{V}_\mathrm{T} = \emptyset$.

Therefore:

$\mathcal{L}(\mathrm{P,CF{-}\lambda,ut}) = \mathcal{L}(\mathrm{P,CF{-}\lambda,ac})$ iff
$\mathcal{L}(\mathrm{P,CF{-}\lambda,ut})$ is closed under right derivatives.

Characterizations of UT

$\mathcal{L}(\mathsf{P},\mathsf{CF}[-\lambda],\mathsf{ut})$ is characterized by:

- $\mathcal{L}(\mathsf{M},\mathsf{CF}[-\lambda],\mathsf{ut})$,
- $\mathcal{L}(\mathsf{rC},\mathsf{CF}[-\lambda],\mathsf{ut})$,
- $\mathcal{L}(\mathsf{TV}_c,\mathsf{CF}[-\lambda],\mathsf{ut})$,
- $\mathcal{L}(\mathsf{G}_c,\mathsf{CF}[-\lambda],\mathsf{ut})$.

TV case quite involved.

Characterizations of AC

$\mathcal{L}$(P,CF[$-\lambda$],ac) is characterized by:

- $\mathcal{L}$(MS,CF[$-\lambda$],ut),
- $\mathcal{L}$(rSC,CF[$-\lambda$],ut),
- $\mathcal{L}$(TV,CF[$-\lambda$],ut),
- $\mathcal{L}$(G,CF[$-\lambda$],ut).

Proof I $\mathcal{L}$(P,CF,ac) $\subseteq$ $\mathcal{L}$(G,CF,ut):

Let $L \in \mathcal{L}(P,CF,ac)$, $L \subseteq V_T^*$,
$L = \bigcup_{a \in V_T} (\{a\} V_T^+ \cap L) \cup ((V_T \cup \{\lambda\}) \cap L)$.
We construct a (G,CF,ut) grammar
$G' = (V_N \cup \{\#, F, S'\}, V_T, P', S', \Gamma, \Sigma, \Phi, h)$ with
$\Gamma = (U, E)$ such that $L(G') = \{a\} V_T^+ \cap L$.
Easy: $\mathcal{L}$(G,CF,ut) is closed under union.

$G = (V_N, V_T, P, S)$ be a (P,CF,ac) grammar generating $\{ w \in V_T^+ \mid aw \in L \}$.

A rule $(p_i : A_i \to w_i, \sigma_i, \phi_i)$ of G is determines two production nodes F_i^+, F_i^- with
$h(F_i^+) = \{A_i \to w_i, \# \to F\}$,
$E \cap \{F_i^+\} \times \{g, r\} \times U = \{F_i^+\} \times \{g, r\} \times (\{F_j^{+/-} \mid p_j \in \sigma_i\} \cup \{t\})$,
$h(F_i^-) = \{A_i \to F\}$,
$E \cap \{F_i^-\} \times \{g, r\} \times U = \{F_i^-\} \times \{g, r\} \times \{F_j^{+/-} \mid p_j \in \phi_i\}$.

Initial node I contains the rule $S' \to \#S$ with green and red arcs leading to $\{F_i^+, F_i^- \mid (p_i : S \to w_i, \sigma_i, \phi_i) \in P\}$.

Final node T contains the rule $\# \to a$ with green and red arcs leading to itself.

Proof II $\mathcal{L}$(G,CF,ut) $\subseteq$ $\mathcal{L}$(TV,CF,ut):

Let $G = (V_N, V_T, P, S, \Gamma, \Sigma, \Phi, h)$ be a (G,CF,ut) grammar with $\Gamma = (U, E)$;
$U = \{U_1, \ldots, U_n\}$,
$U_i = \{A_{i1} \to w_{i1}, \ldots, A_{ir_i} \to w_{ir_i}\}$.

We give a simulating (TV,CF,ut)-grammar
$(V_N \cup \{X^{(i)} \mid X \in V_N, 1 \le i \le n\} \cup \{S', F\} \cup U, P', S', \phi)$, where ϕ is a function with period $2n+3$:
for $i = 1, \ldots, n$, let

$$
\begin{aligned}
\phi(i) &= \{A_{i\rho} \to A_{i\rho}^{(i)} \mid 1 \le \rho \le r_i\} \cup \{U_j \to U_j \mid j \ne i\},\\
\phi(n+i) &= \{A_{j\rho}^{(j)} \to F \mid j \ne i, 1 \le \rho \le r_k\} \cup \{U_j \to U_j \mid j \ne i\};\\
\phi(2n+1) &= \{A_{i\rho}^{(i)} \to w_{i\rho} \mid 1 \le i \le n, 1 \le \rho \le r_i\},\\
\phi(2n+2) &= \{S' \to SU_i \mid U_i \in \Sigma\} \cup \{U_i \to U_j \mid (U_i, g, U_j) \in E\},\\
\phi(2n+3) &= \{U_i \to U_i \mid U_i \in U\} \cup \{U_i \to \lambda \mid U_i \in \Phi\}.
\end{aligned}
$$

Literatur

1. S. Ábrahám. Some questions of phrase-structure grammars I. *Comput. Linguistics*, 4:61–70, 1965.

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

3. H. Fernau. Observations on grammar and language families. Technical Report 22/94, Universität Karlsruhe, Fakultät für Informatik, August 1994. Most of this report will appear in Fundamenta Informaticae in 1996.

4. H. Fernau. A predicate for separating language classes. *EATCS Bulletin*, 56:96–97, June 1995.

5. S. Ginsburg and E. H. Spanier. Control sets on grammars. *Mathematical Systems Theory*, 2:159–177, 1968.

6. D. Hauschildt and M. Jantzen. Petri net algorithms in the theory of matrix grammars. *Acta Informatica*, 31:719–728, 1994.

7. F. Hinz and J. Dassow. An undecidability result for regular languages and its application to regulated rewriting. *EATCS Bulletin*, 38:168–173, 1989.

8. A. Pascu and Gh. Păun. On the planarity of bicolored digraph grammar systems. *Discrete Mathematics*, 25:195–197, 1979.

9. Gh. Păun. Normal forms for bicolored-digraph-grammar systems. *International Journal of Computer Mathematics*, 7:109–118, 1979.

10. D. J. Rosenkrantz. Programmed grammars and classes of formal languages. *Journal of the ACM*, 16(1):107–131, 1969.

11. G. Rozenberg and A. K. Salomaa. Context-free grammars with graph-controlled tables. Technical Report DAIMI PB-43, Institute of Mathematics at the University of Aarhus, Ny Munkegade, DAN–8000 Aarhus C, January 1975.

12. A. K. Salomaa. On grammars with restricted use of productions. *Annales academiae scientarum Fennicae*, Serie A-454:1–32, 1969.

13. A. K. Salomaa. Periodically time-variant context-free grammars. *Information and Control (now Information and Computation)*, 17:294–311, 1970.

14. A. K. Salomaa. *Formal Languages*. Academic Press, 1973.

15. E. D. Stotskii. Control of the conclusion in formal grammars. *Problemy peredachi informacii; translated: Problems of information transmission*, 7(3):257–270, 1971 (Translation 1973).

16. D. Wood. Bicolored digraph grammar systems. *RAIRO Informatique théorique et Applications/Theoretical Informatics and Applications*, 1:45–50, 1973.

17. D. Wood. A note on bicolored digraph grammar systems. *International Journal of Computer Mathematics*, 3:301–308, 1973.