Kontextfreie Grammatiken

Wozu kontextfreie Grammatiken?

Gib einen regulären Ausdruck für folgende Sprache an:

Die Zeichen a, b, c, ... , z gehören zur Sprache

Wenn die Zeichenketten r und s zur Sprache gehören, gehören auch folgende Zeichenketten zur Sprache: $r\,s$ $\quad r|s$ $\quad r*$ $\quad (r)$

Es gibt keinen regulären Ausdruck, der die Sprache beschreibt. Warum?

Endliche Automaten haben nur ein *endliches* Gedächtnis. Um beliebig viele Klammern zählen zu können, brauchen wir aber ein unendliches Gedächtnis.

Kontextfreie Grammatiken: Definition

Eine kontextfreie Grammatik ist ein 4-Tupel (T, N, R, S), wobei

T die Menge der *Terminalsymbole* (der "Eingabezeichen") ist

N die Menge der *Nichtterminalsymbole*, $N \cap T = \emptyset$

R die Menge der *Regeln*, eine Teilmenge von $N \times (N \cup T)*$

S ein ausgewähltes Nichtterminalsymbol, das *Startsymbol*

Beispiel: Eine Grammatik für die Sprache auf der letzten Folie:

$$T = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots, \mathsf{z}, (,), |, * \} \qquad N = \{S\}$$

$$\begin{array}{lll} R: & S \to SS & S \to \mathsf{a} \\ & S \to S \mid S & S \to \mathsf{b} \\ & S \to (S) & \vdots \\ & S \to S* & S \to \mathsf{z} \end{array}$$

Ableitungen mit kontextfreien Grammatiken

Eine Zeichenkette gehört zur von der Grammatik *erzeugten Sprache*, wenn es eine **Ableitung** gibt, die diese Zeichenkette erzeugt.

Was ist eine Ableitung?

- Beginne mit einer Kette, die nur aus dem Startsymbol S besteht und wende sukzessiv Regeln an:
- Nimm ein beliebiges Nichtterminal aus der Kette sowie eine Regel, deren linke Seite dieses Nichtterminal trägt.

  Ersetze das Nichtterminal durch die rechte Seite der Regel. Durch diesen *Ableitungsschritt* erhält man eine neue Kette:

  Aus $w = \alpha A \beta$ und der Regel $A \to \gamma$ wird $w' = \alpha\gamma\beta$ $\quad$ α, β und γ sind aus $(N \cup T)^*$
- Wenn die neue Kette keine Nichtterminale mehr enthält, hat man eine Zeichenkette aus der Sprache abgeleitet

Ableitungen: Beispiel

Die Ableitung von **ab(ba)*** mit der Grammatik von zuvor:

$$
\begin{aligned}
S &\overset{S\to SS}{\Longrightarrow} SS \\
&\overset{S\to \mathsf{a}}{\Longrightarrow} \mathsf{a}S \\
&\overset{S\to SS}{\Longrightarrow} \mathsf{a}SS \\
&\overset{S\to \mathsf{b}}{\Longrightarrow} \mathsf{ab}S \\
&\overset{S\to S*}{\Longrightarrow} \mathsf{ab}S* \\
&\overset{S\to (S)}{\Longrightarrow} \mathsf{ab}(S)* \\
&\overset{S\to SS}{\Longrightarrow} \mathsf{ab}(SS)* \\
&\overset{S\to \mathsf{a}}{\Longrightarrow} \mathsf{ab}(\mathsf{a}S)* \\
&\overset{S\to \mathsf{a}}{\Longrightarrow} \mathsf{ab}(\mathsf{ab})*
\end{aligned}
$$

$$T = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots, \mathsf{z}, (,), |, * \}$$

$$N = \{S\}$$

$$
\begin{array}{lll}
R\!: & S \to SS & S \to \mathsf{a} \\
 & S \to S \mid S & S \to \mathsf{b} \\
 & S \to (S) & \vdots \\
 & S \to S* & S \to \mathsf{z}
\end{array}
$$

Es gibt unter Umständen mehrere äquivalente Ableitungen für eine Zeichenkette!

Ableitungsbäume

Eine Ableitung läßt sich auch als Baumstruktur darstellen, dem **Ableitungsbaum**. Äquivalente Ableitungen bilden auf denselben Ableitungsbaum ab.

Dabei entspricht

- ein innerer Knoten einem ersetzten Nichtterminal
- ein Teilbaum der Tiefe Eins einer Regelanwendung
- die Blätter der Kette, die durch die Ableitung erzeugt wurde
- die Wurzel dem Startknoten

Die Anordnung der Knoten entspricht dabei der Abfolge der Terminale und Nichtterminale in der rechten Seite einer Regel.

Parsing

Parsen bedeutet, einen String aus Terminalsymbolen anhand einer Grammatik zu analysieren, d.h. wenn möglich eine Ableitung zu finden und den Ableitungsbaum zu konstruieren.

Ist die Grammatik für die Analyse einer natürlichen Sprache entworfen, spiegelt der Ableitungsbaum die *Konstituentenstruktur* des zu analysierenden Satzes wider. Das ist normalerweise der erste Schritt zu einer Bedeutungsbestimmung.

Natürliche Sprachen sind typischerweise mehrdeutig, d.h. dass es für einen Satz mehrere Ableitungsbäume geben kann, die unterschiedlichen Bedeutungen entsprechen.

Parsing-Algorithmen

Es gibt verschiedene Parsing-Algorithmen, die sich bezüglich ihrer Verarbeitungsstrategie unterscheiden:

Top-Down (oder zielgesteuert): Beginne mit dem Startsymbol und ersetze so lange Nichtterminale durch rechte Regelseiten, bis die Elemente des Strings erreicht sind (spielt die Ableitung nach)

Bottom-Up (oder datengesteuert): Beginne mit dem String aus Terminalsymbolen und ersetze sie solange mittels der rechten Regelseiten, bis das Startsymbol erreicht ist.

Depth- vs. Breadth-First versuche entweder, so schnell wie möglich einen hohen Baum aufzubauen oder baue zuerst alle kleineren Bäume auf

Uni- oder Bidirektional Suche in eine Richtung (links nach rechts oder umgekehrt) oder unterschiedlich (z.B. je nachdem, wo in einer Regel der linguistische Kopf zu finden ist).

Top-Down Parser

Ein Top-Down Parser geht folgendermaßen vor:

- Starte den Parse mit dem kompletten String als input und dem Startsymbol als Ziel.
- Das derzeitige Ziel ist ein Terminalsymbol:
  - es entspricht dem ersten Zeichen in der Eingabe, dann haben wir dieses Ziel erfolgreich abgeleitet und wir melden den abzuarbeitenden Teil der Eingabe zurück
  - dieses Ziel schlägt fehl
- das derzeitige Ziel ist ein Nichtterminal: versuche nacheinander für alle Regeln, die das Ziel als linke Seite haben, alle Teilziele abzuarbeiten. Zunächst mit der gegebenen Eingabe das erste Teilziel, mit der übriggebliebenen Eingabe das zweite Teilziel und so weiter bis wir alle Teilziele abgeleitet haben. Der Rest der Eingabe, die für die Ableitung der Teilziele nicht gebraucht wurde, wird zurückgegeben.

Ein Top-Down Parser versucht, die Linksableitung einer Kette zu finden bzw. den Parsebaum von der Wurzel aus in Präordnung zu berechnen.

Bottom-Up Parser

Ein Bottom-Up Parser geht folgendermassen vor:

- Suche die Regeln, deren rechte Seite mit einem Teilstring der Eingabe matcht und ersetze den Teilstring.
- Iteriere für die so entstandenen neuen Ketten bis eine Kette gefunden ist, die nur aus dem Startsymbol besteht, oder keine Ersetzungen mehr möglich sind.