Parsebäume

Sei im Folgenden eine Typ-2-Grammatik $G = (V, \Sigma, P, S)$ fixiert.

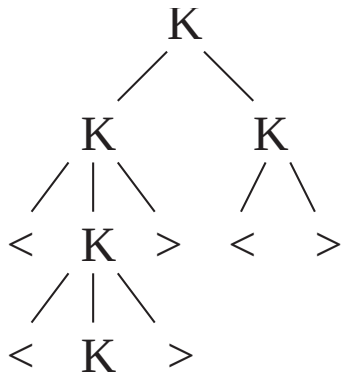
Ein Parsebaum ist ein Baum, dessen Knoten mit Symbolen aus $V \cup \Sigma$ beschriftet sind, mit:

- innere Knoten sind mit $A \in V$ beschriftet
- Blätter sind mit $A \in V$ oder $a \in \Sigma$ beschriftet.
- ist ein Knoten mit $A \in V$ beschriftet, und seine Kinder mit $X_1, \ldots, X_k$, dann ist $A \to X_1 \ldots X_k$ eine Produktion in P.

Das Ergebnis des Parsebaumes ist der String der Symbole an den Blättern, von links nach rechts gelesen.

Parsebäume sind eine sukzinkte Darstellung, wie ein String aus der Grammatik hergeleitet wird, die von unwesentlichen Details wie der Reihenfolge der Ableitungsschritte abstrahiert.

Betrachten Sie als Beispiel die Grammatik aus der letzten Vorlesung für die Sprache L_{kl}. Ein Parsebaum ist z.B.der folgende, mit der Wurzel K und dem Ergebnis $<<K>><>$.



Zur Vereinfachung betrachten wir in diesem Kapitel nur Grammatiken, die Sprachen ohne das Wort ϵ erzeugen.

Linksableitung

Für $\gamma_1, \gamma_2 \in (V \cup \Sigma)^*$ gilt $\gamma_1 \overset{\ell}{\Rightarrow}_G \gamma_2$, falls:

- es gibt eine Produktion $A \to \beta$ in P,
- und Wörter $\delta_1 \in \Sigma^*$ und $\delta_2 \in (V \cup \Sigma)^*$ mit

  $\gamma_1 = \delta_1 A \delta_2$ und $\gamma_2 = \delta_1 \beta \delta_2$.

Eine Folge $\gamma_1 \overset{\ell}{\Rightarrow}_G \gamma_2 \overset{\ell}{\Rightarrow}_G \ldots \overset{\ell}{\Rightarrow}_G \gamma_k$ heißt
Linksableitung von γ_k aus γ_1

$\overset{\ell}{\Rightarrow}^*_G$ ist die reflexive, transitive Hülle von $\overset{\ell}{\Rightarrow}_G$.

Bei einer Linksableitung ist die Wahlfreiheit gegenüber einer allgemeinen Ableitung eingeschränkt: es muss stets die erste in einem String vorkommende Variable ersetzt werden. Als einziger Freiheitsgrad bleibt die Auswahl der verwendeten Produktion.

Eine Linksableitung in der obigen Beispielgrammatik ist die folgende:

$$K \overset{\ell}{\Rightarrow} KK \overset{\ell}{\Rightarrow} <K>K \overset{\ell}{\Rightarrow} <<>>K \overset{\ell}{\Rightarrow} <<>><>$$

Äquivalenz

Für eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$
und Wort $w \in \Sigma^*$ sind äquivalent:

- $w \in L(G)$, also $S \Rightarrow_G^* w$
- $S \overset{\ell}{\Rightarrow}_G^* w$
- Es gibt einen Parsebaum mit Wurzel S und Ergebnis w.

Aus der zweiten Bedingung folgt trivialerweise die Erste, da eine Linksableitung ein Spezialfall einer Ableitung ist

Wir zeigen auf der nächsten Folie, dass die erste Bedingugng die dritte impliziert, und aus der dritten die zweite folgt. Damit ist die Äquivalenz bewiesen.

Beide Implikationen müssen in etwas allgemeinerer Form bewiesen werden, wie in den Lemmas auf der nächsten Folie ausgeführt.

Beweis der Äquivalenz

Lemma

Für $A \in V$ und $\beta \in (V \cup \Sigma)^*$ gilt:

falls $A \Rightarrow_G^* \beta$ gilt, dann gibt es einen Parsebaum mit Wurzel A und Ergebnis β

Lemma

Für $A \in V$ und $w \in \Sigma^*$ gilt:

falls es einen Parsebaum mit Wurzel A und Ergebnis w gibt, dann gilt $A \overset{\ell}{\Rightarrow}_G^* w$.

Beweis des ersten Lemmas durch Induktion nach der Länge der Ableitung von β aus A.

Induktionsanfang: die Länge ist 0, also $\beta = A$. Der Baum aus einem Knoten mit Beschriftung A ist ein Parsebaum mit Wurzel A und Ergebnis A:

Induktionsschritt: die Ableitung hat einen letzten Schritt, also $A \Rightarrow^* \gamma \Rightarrow \beta$. Nach Induktionsvoraussetzung gibt es einen Parsebaum T mit Wurzel A und Ergebnis γ. Ferner ist $\gamma = \delta_1 B \delta_2$ und $\beta = \delta_1 \eta \delta_2$ für eine Produktion $B \to \eta$ in G. Der benötigte Parsebaum entsteht, indem in T unter das Blatt mit Beschriftung B neue Blätter mit den Symbolen aus η gehängt werden.

Beweis des zweiten Lemmas durch Induktion nach der Höhe des Parsebaumes.

Induktionsanfang: der Baum hat die Höhe 1, also sind die Blätter mit Symbolen in w direkt unter der Wurzel. Dann ist aber $A \to w$ eine Produktion in G, und somit $A \overset{\ell}{\Rightarrow} w$.

Induktionsschritt: der Baum hat die Höhe $h+1$. Seien $x_1, \ldots, x_k$ die Beschriftungen der Knoten direkt unter der Wurzel, und für $i = 1 \ldots k$ sei w_i das Ergebnis des Teilbaums mit Wurzel x_i, so dass $w = w_1 \ldots w_k$. Jeder dieser Teilbäume hat Höhe $\leq h$, also gilt nach Induktionshypothese $x_i \overset{\ell}{\Rightarrow}^* w_i$. Diese Linksableitungen können zu einer Linksableitung von w aus A kombiniert werden:

$$A \overset{\ell}{\Rightarrow} x_1 x_2 \ldots x_k \overset{\ell}{\Rightarrow}^* w_1 x_2 \ldots x_k \overset{\ell}{\Rightarrow}^* w_1 w_2 \ldots x_k \overset{\ell}{\Rightarrow}^* \ldots \overset{\ell}{\Rightarrow}^* w_1 w_2 \ldots w_k$$

Motivation

Für $G = (V, \Sigma, P, S)$ und $A \in V$ sei

$$L(G, A) := \{ w \in \Sigma^* ; A \Rightarrow_G^* w \}$$

Kontextfreie Grammatik $\hat{=}$ wechselseitig rekursive Definition der Sprachen $L(G, A)$

Verarbeitung rekursiver Funktionen mit Stack.

$\rightsquigarrow$ Erweiterung endlicher Automaten um einen Stack.

Obwohl Typ 3-Grammatiken ja auch rekursive Definitionen sind, können reguläre Sprachen von einfachen endlichen Automaten ohne Stack erkannt werden. Das liegt daran, dass die Variablen, also der rekursive Aufruf, hier stets am Ende des Definitionsrumpfes steht. Es handelt sich also um *endrekursive* Definitionen, und Endrekursionen können bekanntlich ohne Stack abgearbeitet werden.

Definition

Ein Pushdown-Automat (PDA) A mit Alphabet Σ besteht aus:

- Zustände Q (endlich viele!)
- Stack-Alphabet Γ
- Anfangszustand $q_0 \in Q$
- Anfangssymbol $Z_0 \in \Gamma$
- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$

Kurzschreibweise: $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

Ein PDA ist also ein ϵ-NEA mit einem zusätzlichen Stack. Die Übergangsfunktion liefert zu jedem Zustand, gelesenem Eingabesymbol (oder ϵ) und dem obersten Stacksymbol eine Menge von Paaren, jeweils bestehend aus dem neuen Zustand und einem String aus Stacksymbolen, die das gesehene oberste Stacksymbol ersetzen.

Der folgende PDA dient als Beispiel auf den nächsten Folien. Er hat zwei Zustände $\{q_0, q_1\}$, von denen q_0 der Anfangszustand ist, Alphabet $\{0, 1\}$, und ein zusätzliches Stacksymbol Z, das auch Anfangssymbol ist. Die Übergangsfunktion ist durch die folgenden beiden Tabellen definiert, in denen die Zeilen für den Input, die Spalten für das Stacksymbol stehen:

q_0	Z	0	1
0	$\{(q_0, 0Z)\}$	$\{(q_0, 00)\}$	$\{(q_0, 01)\}$
1	$\{(q_0, 1Z)\}$	$\{(q_0, 10)\}$	$\{(q_0, 11)\}$
ϵ	$\{(q_1, Z)\}$	$\{(q_1, 0)\}$	$\{(q_1, 1)\}$

q_1	Z	0	1
0	$\emptyset$	$\{(q_1, \epsilon)\}$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\{(q_1, \epsilon)\}$
ϵ	$\{(q_1, \epsilon)\}$	$\emptyset$	$\emptyset$

Konfigurationen

Eine Konfiguration eines PDA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$
ist ein Tripel $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$, bestehend aus:

- dem Zustand $q \in Q$
- dem zu lesenden Eingabewort $w \in \Sigma^*$
- dem Stack-Inhalt $\gamma \in \Gamma^*$

Eine Konfiguration ist eine Momentaufnahme der Berechnung eines PDA, in dem sich gemerkt wird, in welchem Zustand der Automat ist, welchen Teil der Eingabe er noch zu verarbeiten hat und was auf dem Stack steht.

Durch eine Konfiguration ist der weitere Verlauf der möglichen Berechnungen eindeutig bestimmt.

Für nichtdeterministische endliche Automaten war ein solcher Begriff nicht nötig, da die weiteren Berechnungen nur vom Zustand abhängen.

Übergangsrelation

Die Übergangsrelation $\vdash_A$ zwischen Konfigurationen eines PDA A ist definiert durch:

- $(q, aw, X\gamma) \vdash_A (p, w, \beta\gamma)$, falls $(p, \beta) \in \delta(q, a, X)$ ist.
- $(q, w, X\gamma) \vdash_A (p, w, \beta\gamma)$, falls $(p, \beta) \in \delta(q, \epsilon, X)$ ist.

$\vdash_A^*$ ist die reflexive, transitive Hülle von $\vdash_A$.

Die folgende Folge von Konfigurationen ist eine mögliche Berechnung des Beispiel-PDA bei Eingabe 0110 von oben:

$$(q_0, 0110, Z) \vdash (q_0, 110, 0Z) \vdash (q_0, 10, 10Z) \vdash (q_1, 10, 10Z)$$
$$\vdash (q_1, 0, 0Z) \vdash (q_1, \epsilon, Z) \vdash (q_1, \epsilon, \epsilon)$$

Akzeptierte Sprache

Der PDA A akzeptiert ein Wort w, wenn er ausgehend von (q_0, w, Z_0) eine Konfiguration erreicht, in der der Stack leer ist:

Definition

Die vom PDA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ akzeptierte Sprache ist:

$$L(A) := \left\{ w \in \Sigma^* \,;\, (q_0, w, Z_0) \vdash_A^* (p, \epsilon, \epsilon) \text{ für ein } p \in Q \right\}$$

Für den Beispiel-PDA oben ist die akzeptierte Sprache $\left\{ ww^R \,;\, w \in \{0,1\} \right\}$.

Um das zu zeigen, sehen wir genauer an, wie der PDA funktioniert.

Im Zustand q_0 kann der PDA jederzeit entweder das gelesene Input-Symbol auf den Stack pushen, oder ohne einen Input zu lesen in den Zustand q_1 wechseln. Im Zustand q_1 kann ein Symbol vom Stack gepoppt werden, wenn es mit dem gelesenen Inputsymbol übereinstimmt. Ist nur noch das Startsymbol auf dem Stack, kann dieses auch entfernt werden, dann wird akzeptiert sofern der input aufgebraucht war.

Das heißt, der PDA errät nichtdeterministisch die Mitte des Strings, schiebt die vordere Hälfte auf den Stack und gleicht sie dann mit der hinteren Hälfte ab. Somit akzeptiert er genau die genannte Sprache.

PDA und kontextfreie Sprachen

Satz

Die von PDA erkannten Sprachen sind genau die kontextfreien Sprachen.

Dazu zeigen wir zwei Teile:

Lemma

Für jede Typ 2-Grammatik G gibt es einen PDA A_G mit $L(A_G) = L(G)$.

Lemma

Für jeden PDA A gibt es eine Typ 2-Grammatik G_A mit $L(G_A) = L(A)$.

Die Äquivalenzen werden wir nicht formal beweisen. Das erste Lemma ist relativ einfach zu beweisen, indem die nach der Konstruktion auf der nächsten Folie stehende Behauptung durch Induktion nach der Länge einer Linksableitung gezeigt wird. Die Konstruktion wird an einem Beispiel verdeutlicht.

So wie der PDA aus dieser Konstruktion funktionieren im wesentlichen auch in der Praxis Parser für kontextfreie Sprachen. Es ist dabei nur dafür zu sorgen, dass der Ablauf des PDA deterministisch gemacht werden kann. Dafür gibt es zahlreiche Methoden, die für verschiedene Teilklassen der kontextfreien Grammatiken funktionieren.

Die umgekehrte Richtung ist schwieriger, aber auch für die Praxis weniger bedeutsam, daher zeigen wir sie hier nicht.

Von der Typ 2-Grammatik zum PDA

Aus $G = (V, \Sigma, P, S)$ definiere den PDA A_G wie folgt:

- Zustände $Q = \{q\}$
- Stack-Alphabet $\Gamma = V \cup \Sigma$
- Anfangszustand q, Anfangssymbol S.
- Für jedes $a \in \Sigma$ ist $(q, \epsilon) \in \delta(q, a, a)$
- Für jede Produktion $A \to \beta$ ist $(q, \beta) \in \delta(q, \epsilon, A)$

Zu zeigen: falls $S \overset{\ell}{\Rightarrow}{}^*_G u\gamma \overset{\ell}{\Rightarrow}{}^*_G w$ für ein u mit $w = uv$,

dann gilt: $(q, w, S) \vdash^*_{A_G} (q, v, \gamma)$

Betrachte als Beispiel die bekannte Grammatik mit den Produktionen $K \to KK$, $K \to <K>$ und $K \to <>$ und Startsymbol K. Der konstruierte PDA hat nur einen Zustand q und das Startsymbol K, sowie die Übergänge:

q	K	$<$	$>$
$<$	$\emptyset$	$\{(q, \epsilon)\}$	$\emptyset$
$>$	$\emptyset$	$\emptyset$	$\{(q, \epsilon)\}$
ϵ	$\{(q, KK), (q, <K>), (q, <>)\}$	$\emptyset$	$\emptyset$

Eine Linksableitung in dieser Grammatik ist

$K \overset{\ell}{\Rightarrow} KK \overset{\ell}{\Rightarrow} <K>K \overset{\ell}{\Rightarrow} <<>>K \overset{\ell}{\Rightarrow} <<>><>$

Dieser entspricht der folgende Ablauf des Automaten, wobei in den Konfigurationen nur der Restinput und Stack angegeben sind:

$$
\begin{array}{lrr}
 & <<>><> & K \\
\vdash & <<>><> & KK \\
\vdash & <<>><> & <K>K \\
\vdash & <>><> & K>K \\
\vdash & <>><> & <>>K \\
\vdash & >><> & >>K \\
\vdash^* & <> & K \\
\vdash & <> & <> \\
\vdash^* & \epsilon & \epsilon
\end{array}
$$