~~CFG~~ kontextfreie Grammatiken

BC George (FH Bielefeld)

Wiederholung

Endliche Automaten, reguläre Ausdrücke, reguläre Grammatiken, reguläre Sprachen

- Wie sind DFAs und NFAs definiert?
- Was sind reguläre Ausdrücke?
- Was sind formale und reguläre Grammatiken?
- In welchem Zusammenhang stehen all diese Begriffe?
- Wie werden DFAs und reguläre Ausdrücke im Compilerbau eingesetzt?

Motivation

Wofür reichen reguläre Sprachen nicht?

Für z. B. alle Sprachen, in deren Wörtern Zeichen über eine Konstante hinaus gezählt werden müssen. Diese Sprachen lassen sich oft mit Variablen im Exponenten beschreiben, die unendlich viele Werte annehmen können.

- $a^i b^{2*i}$ ist nicht regulär — kein regex
- $a^i b^{2*i}$ für $0 \leq i \leq 3$ ist regulär, äquivalenter regex: $\varepsilon|abb|aabbbb|aaabbbbbb$

// Die as und bs müssen gemacht werden

- Wo finden sich die oben genannten VAriablen bei einem DFA wieder?
- Warum ist die erste Sprache oben nicht regulär, die zweite aber?

machen in Programmiersprachen

if $((a+b)*c-d)\cdot e < (a+d)\cdot b)$

Klammern matchen!

Themen für heute

- PDAs: mächtiger als DFAs, NFAs
- kontextfreie Grammatiken und Sprachen: mächtiger als reguläre Grammatiken und Sprachen
- DPDAs und deterministisch kontextfreie Grammatiken: die Grundlage der Syntaxanalyse im Compilerbau

Einordnung: Erweiterung der Automatenklasse DFA, um komplexere Sprachen als die regulären akzeptieren zu können

Wir spendieren den DFAs einen möglichst einfachen, aber beliebig großen, Speicher, um zählen und matchen zu können. Wir suchen dabei konzeptionell die "kleinstmögliche" Erweiterung, die die akzeptierte Sprachklasse gegenüber DFAs vergrößert.

- Der konzeptionell einfachste Speicher ist ein Stack. Wir haben keinen wahlfreien Zugriff auf die gespeicherten Werte.
- Es soll eine deterministische und eine indeterministische Variante der neuen Automatenklasse geben.
- In diesem Zusammenhang wird der Stack auch Keller genannt.

Kellerautomaten (Push-Down-Automata, PDAs)

Def.: Ein Kellerautomat (PDA) $P = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ \perp,\ F)$ ist ein Septupel mit:

Q:	eine endliche Menge von Zuständen
Σ:	eine endliche Menge von Eingabesymbolen
Γ:	ein endliches Kelleralphabet
δ:	die Übergangsfunktion $\delta : Q \times \Sigma \cup \{\epsilon\} \times \Gamma \to 2^{Q \times \Gamma^*}$
q_0:	der Startzustand
$\perp \in \Gamma$:	der anfängliche Kellerinhalt, symbolisiert den leeren Keller ($\perp$ = bottom)
$F \subseteq Q$:	die Menge von Endzuständen

Abbildung 1: Definition eines PDAs

Ein PDA ist per Definition nichtdeterministisch und kann spontane Zustandsübergänge durchführen.

δ : Zustand + Eingabezeichen (evtl. ϵ) + oberstes Kellerelement liefert eine Menge von Paaren aus Zustand + zu pushende Symbole

Was kann man damit akzeptieren?

+ das aktuelle Zustand

Strukturen mit paarweise zu matchenden Symbolen.

Bei jedem Zustandsübergang wird ein Zeichen (oder ϵ) aus der Eingabe gelesen, ein Symbol von Keller genommen. Diese und das Eingabezeichen bestimmen den Folgezustand und eine Zeichenfolge, die auf den Stack gepackt wird. Dabei wird ein Symbol, das später mit einem Eingabesymbol zu matchen ist, auf den Stack gepackt.

Soll das automatisch vom Stack genommene Symbol auf dem Stack bleiben, muss es wieder gepusht werden.

Beispiel

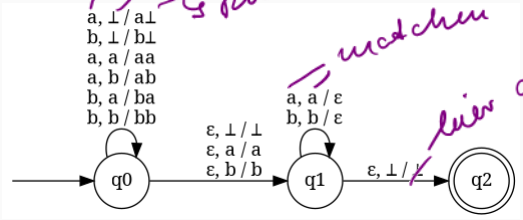


Abbildung 2: Ein PDA für $L = \{ww^R \mid w \in \{a, b\}^*\}$

Konfigurationen von PDAs

Def.: Eine Konfiguration (ID) eines PDAs 3-Tupel (q, w, γ) mit

- q ist ein Zustand
- w ist der verbleibende Input, $w \in \Sigma^*$
- γ ist der Kellerinhalt $\gamma \in \Gamma^*$

eines PDAs zu einem gegebenen Zeitpunkt.

z.B. der Automat oben:

$(q_1, abb, abb\$)$

↑ nächster ↑ oben

ursprüngliche Eingabe

bbaabb

Die Übergangsrelation eines PDAs

Def.: Die Relation $\vdash$ definiert Übergänge von einer Konfiguration zu einer anderen:

Sei $(p, \alpha) \in \delta(q, a, X)$, dann gilt $\forall w \in \Sigma^*$ und $\beta \in \Gamma^*$:

$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$.

Def.: Wir definieren mit $\overset{*}{\vdash}$ 0 oder endlich viele Schritte des PDAs induktiv wie folgt:

- Basis: $I \overset{*}{\vdash} I$ für eine ID I.
- Induktion: $I \overset{*}{\vdash} J$, wenn $\exists$ ID K mit $I \vdash K$ und $K \overset{*}{\vdash} J$.

Eigenschaften der Konfigurationsübergänge

Satz: Sei $P = (Q, \Sigma, \Gamma, \delta, q_0, \bot, F)$ ein PDA und $(q, x, \alpha) \overset{*}{\vdash} (p, y, \beta)$. Dann gilt für beliebige Strings $w \in \Sigma^*$, γ in Γ^*:

$(q, xw, \alpha\gamma) \overset{*}{\vdash} (p, yw, \beta\gamma)$

Satz: Sei $P = (Q, \Sigma, \Gamma, \gamma, q_0, \bot, F)$ ein PDA und $(q, xw, \alpha) \overset{*}{\vdash} (p, yw, \beta)$.

Dann gilt: $(q, x, a) \overset{*}{\vdash} (p, y, \beta)$

Akzeptierte Sprachen

gesamte Restkonfig

Def.: Sei $P = (Q, \Sigma, \Gamma, \delta, q_0, \bot, F)$ ein PDA. Dann ist die *über einen Endzustand* akzeptierte Sprache $L(P) = \{w \mid (q_0, w, \bot) \overset{*}{\vdash} (q, \epsilon, \alpha)\}$ für einen Zustand $q \in F, \alpha \in \Gamma^*$.

1D

Def.: Für einen PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \bot, F)$ definieren wir die über den *leeren Keller* akzeptierte Sprache $N(P) = \{(w \mid (q_0, w, \bot) \overset{*}{\vdash} (q, \epsilon, \epsilon)\}$.

Akzeptanzäquivalenzen

Satz: Wenn $L = N(P_N)$ für einen PDA P_N, dann gibt es einen PDA P_L mit $L = L(P_L)$.

Satz: Für einen PDA P mit ϵ-Transitionen existiert ein PDA Q ohne ϵ-Transitionen mit $L(P) = N(P) = L(Q) = N(Q)$.

Die Transitionsfunktion δ ist dann von der Form $\delta : Q \times \Sigma \times \Gamma \to 2^{Q \times \Gamma^*}$.

Deterministische PDAs

Def. Ein PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \bot, F)$ ist *deterministisch* $:\Leftrightarrow$

- $\delta(q, a, X)$ hat höchstens ein Element für jedes $q \in Q, a \in \Sigma$ oder ($a = \epsilon$ und $X \in \Gamma$).
- Wenn $\delta(q, a, X)$ nicht leer ist für ein $a \in \Sigma$, dann muss $\delta(q, \epsilon, X)$ leer sein.

Deterministische PDAs werden auch *DPDAs* genannt.

$\delta(q_1, a, X)$ und $\delta(q_1, \epsilon, X)$ geht nicht beides

Der kleine Unterschied

Satz: Die von DPDAs akzeptierten Sprachen sind eine echte Teilmenge der von PDAs akzeptierten Sprachen.

Die Sprachen, die von *regex* beschrieben werden, sind eine echte Teilmenge der von DPDAs akzeptierten Sprachen.

Kontextfreie Grammatiken und Sprachen

Kontextfreie Grammatiken

Def. Eine *kontextfreie (cf-)* Grammatik ist ein 4-Tupel $G = (N, T, P, S)$ mit N, T, S wie in (formalen) Grammatiken und P ist eine endliche Menge von Produktionen der Form:

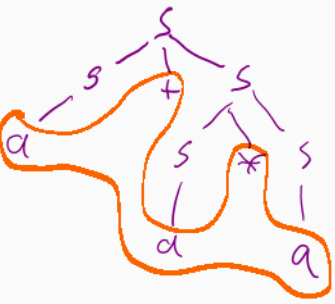
$X \to Y$ mit $X \in N, Y \in (N \cup T)^*$.

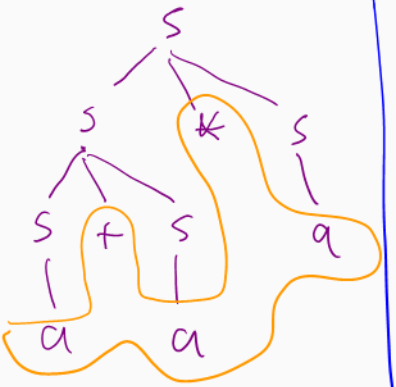
$\Rightarrow, \overset{*}{\Rightarrow}$ sind definiert wie bei regulären Sprachen. Bei cf-Grammatiken nennt man die Ableitungsbäume oft *Parse trees*.

Beispiel

$S \to a \mid S + S \mid S * S$

Ableitungsbäume für $a + a * a$:





Das können wir nicht gebrauchen!

Achtung:

$S \Rightarrow ABC \mid \ldots$

$A \Rightarrow \alpha \mid \ldots$

$B \Rightarrow \beta \mid \ldots$

$C \Rightarrow \gamma \mid \ldots$

Es spielt keine Rolle, ob man hier mit A, B oder C weitermacht, die Struktur bleibt gleich.

Mehrdeutige Grammatiken

siehe Beispiel oben

Def.: Gibt es in einer von einer kontextfreien Grammatik erzeugten Sprache ein Wort, für das mehr als ein Ableitungsbaum existiert, so heißt diese Grammatik *mehrdeutig*. Anderenfalls heißt sie *eindeutig*.

Satz: Es gibt kontextfreie Sprachen, für die keine eindeutige Grammatik existiert.

Kontextfreie Grammatiken und PDAs

Satz: Die kontextfreien Sprachen und die Sprachen, die von PDAs akzeptiert werden, sind dieselbe Sprachklasse.

Akzeptanz über leeren Keller

Satz: Sei $L = N(P)$ für einen DPDA P, dann hat L eine eindeutige Grammatik.

Def.: Die Klasse der Sprachen, die von einem DPDA akzeptiert werden, heißt Klasse der *deterministisch kontextfreien (oder LR(k)-) Sprachen*.

Das Pumping Lemma für kontextfreie Sprachen

Wenn wir beweisen müssen, dass eine Sprache nicht cf ist, hilft das Pumping Lemma für cf-Sprachen:

Satz: Sei L eine kontextfreie Sprache

Wenn p nicht existiert, ist die Sprache nicht cf.

$\Rightarrow \exists$ eine Konstante $p \in \mathbb{N}$:

$\forall_{\substack{z \in L \\ |z| \geq p}} \exists\ u, v, w, x, y \in \Sigma^*$ mit $z = uvwxy$ und

- $| vwx | \leq p$
- $vx \neq \epsilon$
- $\forall i \geq 0 : uv^i wx^i y \in L$

Es darf auch noch andere Teile im Wort geben, die man aufpumpen könnte

die v und x zu matchenden Teile

Abschlusseigenschaften von kontextfreien Sprachen

Satz: Die kontextfreien Sprachen sind abgeschlossen unter:

- Vereinigung
- Konkatenation
- Kleene-Hüllen L^* und L^+

Satz: Wenn L kontextfrei ist, dann ist L^R kontextfrei.

Entscheidbarkeit von kontextfreien Grammatiken und Sprachen

Satz: Es ist entscheidbar für eine kontextfreie Grammatik G,

- ob $L(G) = \emptyset$
- welche Symbole nach ϵ abgeleitet werden können
- welche Symbole erreichbar sind
- ob $w \in L(G)$ für ein gegebenes $w \in \Sigma^*$

Satz: Es ist nicht entscheidbar,

- ob eine gegebene kontextfreie Grammatik eindeutig ist
- ob der Durchschnitt zweier kontextfreier Sprachen leer ist
- ob zwei kontextfreie Sprachen identisch sind
- ob eine gegebene kontextfreie Sprache gleich Σ^* ist

Abschlusseigenschaften deterministisch kontextfreier Sprachen

Satz: Deterministisch kontextfreie Sprachen sind abgeschlossen unter

- Durchschnitt mit regulären Sprachen
- Komplement

Sie sind nicht abgeschlossen unter

- Umkehrung
- Vereinigung
- Konkatenation

Wrap-Up

Das sollen Sie mitnehmen

- Die Struktur von gängigen Programmiersprachen lässt sich nicht mit regulären Ausdrücken beschreiben und damit nicht mit DFAs akzeptieren.
- Das Automatenmodell der DFAs wird um einen endlosen Stack erweitert, das ergibt PDAs.
- Kontextfreie Grammatiken (CFGs) erweitern die regulären Grammatiken.
- Deterministisch parsebare Sprachen haben eine eindeutige kontextfreie Grammatik.
- Es ist nicht entscheidbar, ob eine gegebene kontextfreie Grammatik eindeutig ist.

Cfg sind ohne weitere Einschränkungen nicht für den Compilerbau geeignet.

LICENSE