CFG

context free grammars
Kontext-freie Grammatiken

BC George (HSBI)

Wiederholung

Endliche Automaten, reguläre Ausdrücke, reguläre Grammatiken, reguläre Sprachen

- Was ist ein Lexer?
- In welchem Zusammenhang stehen Lexer und reguläre Sprachen?
- Was können Lexer nicht?

Motivation

Was brauchen wir jetzt?

„reguläre" Sprache zur Erkennung der Syntax

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
- Syntaxanalyse

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) $P = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ \bot,\ F)$ 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
$\bot \in \Gamma$:	der anfängliche Kellerinhalt, symbolisiert den leeren Keller ($\bot$ = bottom)
$F \subseteq Q$:	die Menge von Endzuständen

Abbildung 1: Definition eines PDAs

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

Was kann man damit akzeptieren?

$a^i b^i c^i$

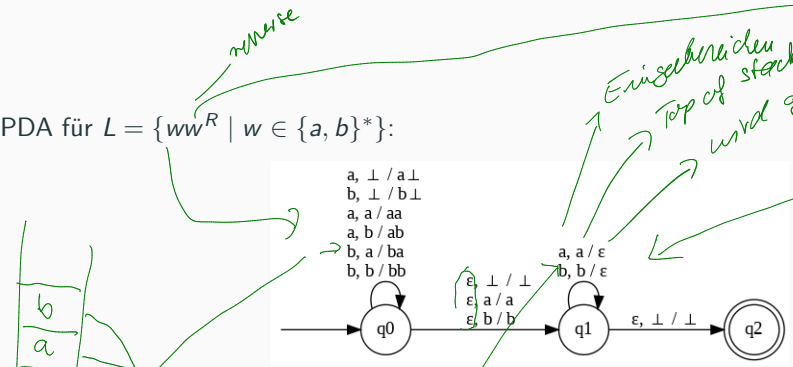
Strukturen mit paarweise zu matchenden Symbolen.

pop

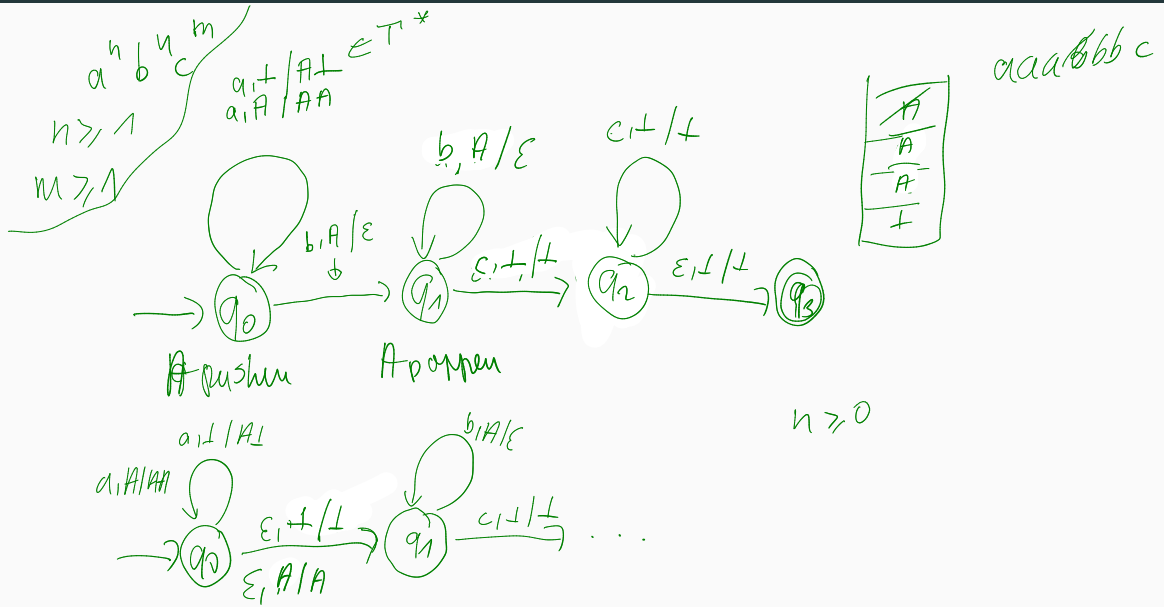
Bei jedem Zustandsübergang wird ein Zeichen (oder ϵ) aus der Eingabe gelesen, ein Symbol von Keller genommen. Diese und das Eingabezeichen bestimmen den Folgezustand und eine Zeichenfolge, die auf den Stack gepackt wird. Dabei wird ein Symbol, (z. B. eines, das später mit einem Eingabesymbol zu matchen ist,) auf den Stack gepackt. Soll das automatisch vom Stack genommene Symbol auf dem Stack bleiben, muss es wieder gepusht werden.

Beispiel

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



Noch ein Beispiel



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.

Der kleine Unterschied

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

Die regulären Sprachen sind eine echte Teilmenge der von DPDAs akzeptierten Sprachen.

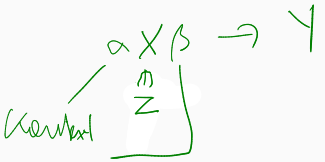
Kontextfreie Grammatiken und Sprachen

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

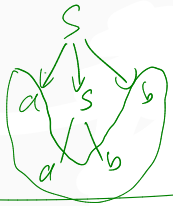
$\Rightarrow, \overset{*}{\Rightarrow}$ sind definiert wie bei regulären Sprachen. Bei cf-Grammatiken nennt man die Ableitungsbäume oft *Parse trees*.



Beispiel

$a^n b^n$

$S \to aSb \mid ab$



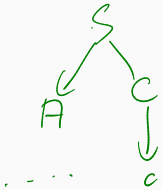
cf

$a^n b^n c^m$

$S \to AC$

$C \to cC \mid c$

$A \to aAb \mid ab$



cf

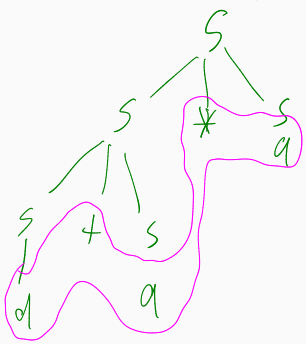
ein Nichtterminal

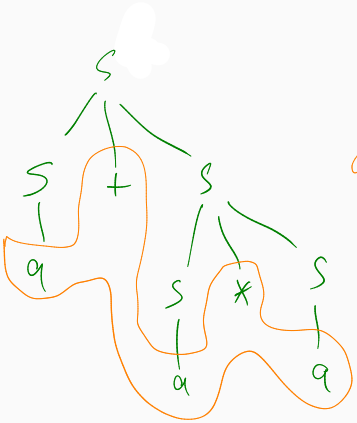
$(N \cup T)^*$

Was ist hier los?

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

Ableitungsbäume für $a + a * a$:





$a + (a * a)$

Nicht jede kontextfreie Grammatik ist eindeutig

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 ist nicht entscheidbar, ob eine gegebene kontextfreie Grammatik eindeutig ist.

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

es gibt keinen Algorithmus

Kontextfreie Grammatiken und PDAs

REG = DFA
CF = PDA

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

Satz: Eine von einem DPDA akzeptierte Sprache hat eine eindeutige Grammatik, aber nicht auch umgekehrt.

Vorgehensweise im Compilerbau: Eine (cf) Grammatik für die gewünschte Sprache definieren und schauen, ob sich daraus ein DPDA generieren lässt (automatisch).

$S \to ABC \mid S \to DBC$
$A \to D$

$S \to B \mid C$

Syntaxanalyse

Syntax

Wir verstehen unter Syntax eine Menge von Regeln, die die Struktur von Daten (z. B. Programmen) bestimmen.

Ziele der Syntaxanalyse

$a = 5$
$\rightarrow a = \text{"abc"}$

- Bestimmung der syntaktischen Struktur eines Programms
- aussagekräftige Fehlermeldungen, wenn ein Eingabeprogramm syntaktisch nicht korrekt ist
- Erstellung des AST (abstrakter Syntaxbaum): Der Parse Tree ohne Symbole, die nach der Syntaxanalyse inhaltlich irrelevant sind (z. B. Semikolons, manche Schlüsselwörter)
- die Symboltabelle(n) mit Informationen bzgl. Bezeichner (Variable, Funktionen und Methoden, Klassen, benutzerdefinierte Typen, Parameter, ...), aber auch die Gültigkeitsbereiche

Was brauchen wir für die Syntaxanalyse von Programmen?

- einen Grammatiktypen, aus dem sich manuell oder automatisiert ein Programm zur deterministischen Syntaxanalyse (= Parser) erstellen lässt
- einen Algorithmus zum Parsen von Programmen mit Hilfe einer solchen Grammatik

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.
- PDAs akzeptieren kontextfreie Sprachen.
- Deterministisch parsbare Sprachen haben eine eindeutige kontextfreie Grammatik, aber nicht für jede eindeutige kontextfreie Grammatik lässt sich ein deterministischer PDA finden.
- Es ist nicht entscheidbar, ob eine gegebene kontextfreie Grammatik eindeutig ist.
- Syntaxanalyse wird mit (möglichst deterministisch) kontextfreien Grammatiken durchgeführt.
- In der Praxis werden aus kontextfreien Grammatiken Parser automatisch generiert.