Foundations of Informatics: a Bridging Course

Week 3: Formal Languages and Processes
Part B: Context-Free Languages
b-it Bonn; 02–06 March 2020

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ws-19-20/foi/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline of Part B

Introductory Example I

Example B.1

Syntax definition of programming languages by “Backus-Naur” rules

Here: simple arithmetic expressions

$$
\begin{array}{rcl}
\langle \mathit{Expression} \rangle & ::= & 0 \\
& \mid & 1 \\
& \mid & \langle \mathit{Expression} \rangle + \langle \mathit{Expression} \rangle \\
& \mid & \langle \mathit{Expression} \rangle * \langle \mathit{Expression} \rangle \\
& \mid & (\langle \mathit{Expression} \rangle)
\end{array}
$$

Meaning:

An expression is either 0 or 1, or it is of the form $u + v$, $u * v$, *or* (u) *where* u, v *are again expressions*

Introductory Example II

Example B.1 (continued)

Here we abbreviate $\langle \mathit{Expression} \rangle$ as E, and use "$\to$" instead of "$::=$". Thus:

$$E \ \to \ 0 \mid 1 \mid E + E \mid E * E \mid (E)$$

Introductory Example II

Example B.1 (continued)

Here we abbreviate $\langle Expression \rangle$ as E, and use "$\rightarrow$" instead of "$::=$". Thus:

$$E \rightarrow 0 \mid 1 \mid E + E \mid E * E \mid (E)$$

Now expressions can be generated by replacing nonterminal symbols according to rules, beginning with the start symbol E:

$$E \Rightarrow E * E$$

Example B.1 (continued)

Here we abbreviate $\langle \mathit{Expression} \rangle$ as E, and use "$\rightarrow$" instead of "$::=$". Thus:

$$E \rightarrow 0 \mid 1 \mid E + E \mid E * E \mid (E)$$

Now expressions can be generated by replacing nonterminal symbols according to rules, beginning with the start symbol E:

$$\begin{aligned} E &\Rightarrow E * E \\ &\Rightarrow (E) * E \end{aligned}$$

Introductory Example II

Example B.1 (continued)

Here we abbreviate $\langle Expression \rangle$ as E, and use "$\to$" instead of "$::=$". Thus:

$$E \to 0 \mid 1 \mid E + E \mid E * E \mid (E)$$

Now expressions can be generated by replacing nonterminal symbols according to rules, beginning with the start symbol E:

$$\begin{aligned} E &\Rightarrow E * E \\ &\Rightarrow (E) * E \\ &\Rightarrow (E) * 1 \end{aligned}$$

Introductory Example II

Example B.1 (continued)

Here we abbreviate $\langle Expression \rangle$ as E, and use "$\to$" instead of "$::=$". Thus:

$$E \to 0 \mid 1 \mid E + E \mid E * E \mid (E)$$

Now expressions can be generated by replacing nonterminal symbols according to rules, beginning with the start symbol E:

$$\begin{aligned} E &\Rightarrow E * E \\ &\Rightarrow (E) * E \\ &\Rightarrow (E) * 1 \\ &\Rightarrow (E + E) * 1 \end{aligned}$$

Introductory Example II

Example B.1 (continued)

Here we abbreviate $\langle Expression \rangle$ as E, and use "$\rightarrow$" instead of "$::=$". Thus:

$$E \rightarrow 0 \mid 1 \mid E + E \mid E * E \mid (E)$$

Now expressions can be generated by replacing nonterminal symbols according to rules, beginning with the start symbol E:

$$\begin{aligned} E &\Rightarrow E * E \\ &\Rightarrow (E) * E \\ &\Rightarrow (E) * 1 \\ &\Rightarrow (E + E) * 1 \\ &\Rightarrow (0 + E) * 1 \end{aligned}$$

Introductory Example II

Example B.1 (continued)

Here we abbreviate $\langle Expression \rangle$ as E, and use "$\to$" instead of "$::=$". Thus:

$$E \to 0 \mid 1 \mid E + E \mid E * E \mid (E)$$

Now expressions can be generated by replacing nonterminal symbols according to rules, beginning with the start symbol E:

$$\begin{aligned} E &\Rightarrow E * E \\ &\Rightarrow (E) * E \\ &\Rightarrow (E) * 1 \\ &\Rightarrow (E + E) * 1 \\ &\Rightarrow (0 + E) * 1 \\ &\Rightarrow (0 + 1) * 1 \end{aligned}$$

Context-Free Grammars I

Definition B.2

A context-free grammar (CFG) is a quadruple

$$G = \langle N, \Sigma, P, S \rangle$$

where

- N is a finite set of nonterminal symbols
- Σ is the (finite) alphabet of terminal symbols (disjoint from N)
- P is a finite set of production rules of the form $A \to \alpha$ where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
- $S \in N$ is a start symbol

Context-Free Grammars II

Example B.3

For the above example, we have:

- $N = \{E\}$
- $\Sigma = \{0, 1, +, *, (,)\}$
- $P = \{E \to 0, E \to 1, E \to E + E, E \to E * E, E \to (E)\}$
- $S = E$

Context-Free Grammars II

Example B.3

For the above example, we have:

- $N = \{E\}$
- $\Sigma = \{0, 1, +, *, (,)\}$
- $P = \{E \to 0, E \to 1, E \to E + E, E \to E * E, E \to (E)\}$
- $S = E$

Naming conventions:

- nonterminals start with uppercase letters
- terminals start with lowercase letters
- start symbol = symbol on LHS of first production

$\Rightarrow$ grammar completely defined by productions

Context-Free Languages I

Definition B.4

Let $G = \langle N, \Sigma, P, S \rangle$ be a CFG.

- A sentence $\gamma \in (N \cup \Sigma)^*$ is directly derivable from $\beta \in (N \cup \Sigma)^*$ if there exist $\pi = A \to \alpha \in P$ and $\delta_1, \delta_2 \in (N \cup \Sigma)^*$ such that $\beta = \delta_1 A \delta_2$ and $\gamma = \delta_1 \alpha \delta_2$ (notation: $\beta \stackrel{\pi}{\Rightarrow} \gamma$ or just $\beta \Rightarrow \gamma$).

Context-Free Languages I

Definition B.4

Let $G = \langle N, \Sigma, P, S \rangle$ be a CFG.

- A sentence $\gamma \in (N \cup \Sigma)^*$ is directly derivable from $\beta \in (N \cup \Sigma)^*$ if there exist $\pi = A \to \alpha \in P$ and $\delta_1, \delta_2 \in (N \cup \Sigma)^*$ such that $\beta = \delta_1 A \delta_2$ and $\gamma = \delta_1 \alpha \delta_2$ (notation: $\beta \overset{\pi}{\Rightarrow} \gamma$ or just $\beta \Rightarrow \gamma$).
- A derivation (of length $n \in \mathbb{N}$) of γ from β is a sequence of direct derivations of the form $\delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_n$ where $\delta_0 = \beta$, $\delta_n = \gamma$, and $\delta_{i-1} \Rightarrow \delta_i$ for every $i \in \{1, \ldots, n\}$ (notation: $\beta \Rightarrow^* \gamma$).

Definition B.4

Let $G = \langle N, \Sigma, P, S\rangle$ be a CFG.

- A sentence $\gamma \in (N \cup \Sigma)^*$ is directly derivable from $\beta \in (N \cup \Sigma)^*$ if there exist $\pi = A \to \alpha \in P$ and $\delta_1, \delta_2 \in (N \cup \Sigma)^*$ such that $\beta = \delta_1 A \delta_2$ and $\gamma = \delta_1 \alpha \delta_2$ (notation: $\beta \stackrel{\pi}{\Rightarrow} \gamma$ or just $\beta \Rightarrow \gamma$).
- A derivation (of length $n \in \mathbb{N}$) of γ from β is a sequence of direct derivations of the form $\delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_n$ where $\delta_0 = \beta$, $\delta_n = \gamma$, and $\delta_{i-1} \Rightarrow \delta_i$ for every $i \in \{1, \ldots, n\}$ (notation: $\beta \Rightarrow^* \gamma$).
- A word $w \in \Sigma^*$ is called derivable in G if $S \Rightarrow^* w$.

Context-Free Languages I

Definition B.4

Let $G = \langle N, \Sigma, P, S \rangle$ be a CFG.

- A sentence $\gamma \in (N \cup \Sigma)^*$ is directly derivable from $\beta \in (N \cup \Sigma)^*$ if there exist $\pi = A \to \alpha \in P$ and $\delta_1, \delta_2 \in (N \cup \Sigma)^*$ such that $\beta = \delta_1 A \delta_2$ and $\gamma = \delta_1 \alpha \delta_2$ (notation: $\beta \stackrel{\pi}{\Rightarrow} \gamma$ or just $\beta \Rightarrow \gamma$).
- A derivation (of length $n \in \mathbb{N}$) of γ from β is a sequence of direct derivations of the form $\delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_n$ where $\delta_0 = \beta$, $\delta_n = \gamma$, and $\delta_{i-1} \Rightarrow \delta_i$ for every $i \in \{1, \ldots, n\}$ (notation: $\beta \Rightarrow^* \gamma$).
- A word $w \in \Sigma^*$ is called derivable in G if $S \Rightarrow^* w$.
- The language generated by G is $L(G) := \{w \in \Sigma^* \mid S \Rightarrow^* w\}$.

Context-Free Languages I

Definition B.4

Let $G = \langle N, \Sigma, P, S \rangle$ be a CFG.

- A sentence $\gamma \in (N \cup \Sigma)^*$ is directly derivable from $\beta \in (N \cup \Sigma)^*$ if there exist $\pi = A \to \alpha \in P$ and $\delta_1, \delta_2 \in (N \cup \Sigma)^*$ such that $\beta = \delta_1 A \delta_2$ and $\gamma = \delta_1 \alpha \delta_2$ (notation: $\beta \stackrel{\pi}{\Rightarrow} \gamma$ or just $\beta \Rightarrow \gamma$).
- A derivation (of length $n \in \mathbb{N}$) of γ from β is a sequence of direct derivations of the form $\delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_n$ where $\delta_0 = \beta$, $\delta_n = \gamma$, and $\delta_{i-1} \Rightarrow \delta_i$ for every $i \in \{1, \ldots, n\}$ (notation: $\beta \Rightarrow^* \gamma$).
- A word $w \in \Sigma^*$ is called derivable in G if $S \Rightarrow^* w$.
- The language generated by G is $L(G) := \{w \in \Sigma^* \mid S \Rightarrow^* w\}$.
- A language $L \subseteq \Sigma^*$ is called context-free (CFL) if it is generated by some CFG.

Context-Free Languages I

Definition B.4

Let $G = \langle N, \Sigma, P, S \rangle$ be a CFG.

- A sentence $\gamma \in (N \cup \Sigma)^*$ is directly derivable from $\beta \in (N \cup \Sigma)^*$ if there exist $\pi = A \to \alpha \in P$ and $\delta_1, \delta_2 \in (N \cup \Sigma)^*$ such that $\beta = \delta_1 A \delta_2$ and $\gamma = \delta_1 \alpha \delta_2$ (notation: $\beta \stackrel{\pi}{\Rightarrow} \gamma$ or just $\beta \Rightarrow \gamma$).
- A derivation (of length $n \in \mathbb{N}$) of γ from β is a sequence of direct derivations of the form $\delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_n$ where $\delta_0 = \beta$, $\delta_n = \gamma$, and $\delta_{i-1} \Rightarrow \delta_i$ for every $i \in \{1, \ldots, n\}$ (notation: $\beta \Rightarrow^* \gamma$).
- A word $w \in \Sigma^*$ is called derivable in G if $S \Rightarrow^* w$.
- The language generated by G is $L(G) := \{w \in \Sigma^* \mid S \Rightarrow^* w\}$.
- A language $L \subseteq \Sigma^*$ is called context-free (CFL) if it is generated by some CFG.
- Two grammars G_1, G_2 are equivalent if $L(G_1) = L(G_2)$.

Context-Free Languages II

Example B.5

The language $\{a^n b^n \mid n \in \mathbb{N}\}$ is context-free. It is generated by the grammar $G = \langle N, \Sigma, P, S \rangle$ with

- $N = \{S\}$
- $\Sigma = \{a, b\}$
- $P = \{S \to aSb \mid \varepsilon\}$

(proof: generating $a^n b^n$ requires exactly n applications of the first and one concluding application of the second rule)

Context-Free Languages II

Example B.5

The language $\{a^n b^n \mid n \in \mathbb{N}\}$ is context-free. It is generated by the grammar $G = \langle N, \Sigma, P, S \rangle$ with

- $N = \{S\}$
- $\Sigma = \{a, b\}$
- $P = \{S \to aSb \mid \varepsilon\}$

(proof: generating $a^n b^n$ requires exactly n applications of the first and one concluding application of the second rule)

Remark: illustration of derivations by derivation trees

- root labelled by start symbol
- leaves labelled by terminal symbols
- successors of node labelled according to right-hand side of production rule
- sequence of leaf symbols = generated word

Summary: Context-Free Grammars and Languages

Seen:

- Context-free grammars
- Derivations
- Context-free languages

Summary: Context-Free Grammars and Languages

Seen:

- Context-free grammars
- Derivations
- Context-free languages

Next:

- Relation between context-free and regular languages

Outline of Part B

Context-Free vs. Regular Languages

Theorem B.6

1. *Every regular language is context-free.*
2. *There exist CFLs which are not regular.*

(Thus: regular languages are a proper subset of CFLs.)

Context-Free vs. Regular Languages

Theorem B.6

1. *Every regular language is context-free.*
2. *There exist CFLs which are not regular.*

(Thus: regular languages are a proper subset of CFLs.)

Proof.

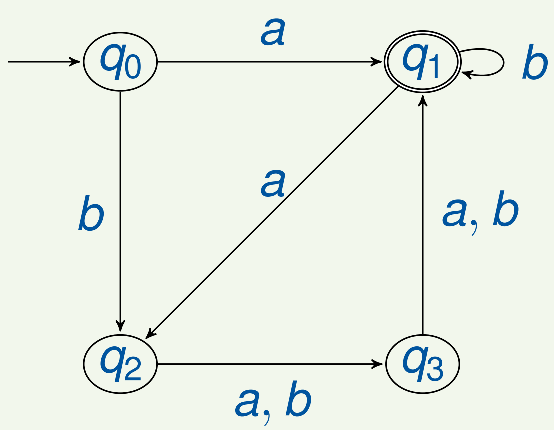
1. Let L be a regular language, and let $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ be a DFA which recognises L.
   $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ is defined as follows:
   - $N := Q$, $S := q_0$
   - if $\delta(q, a) = q'$, then $q \to aq' \in P$
   - if $q \in F$, then $q \to \varepsilon \in P$

   Obviously a w-labelled run in $\mathfrak{A}$ from q_0 to F corresponds to a derivation of w in $G_{\mathfrak{A}}$, and vice versa. Thus $L(\mathfrak{A}) = L(G_{\mathfrak{A}})$ (example on the following slide).
2. An example is $\{a^n b^n \mid n \in \mathbb{N}\}$ (see Ex. B.5).
   Intuitive reason: recognising this language requires “unbounded counting” capability. □

From Regular to Context-Free Languages

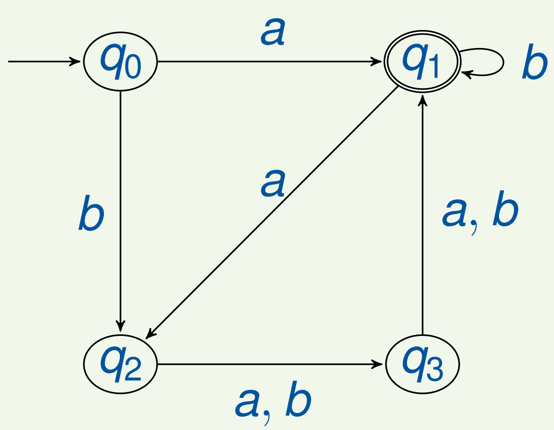
Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:



Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:

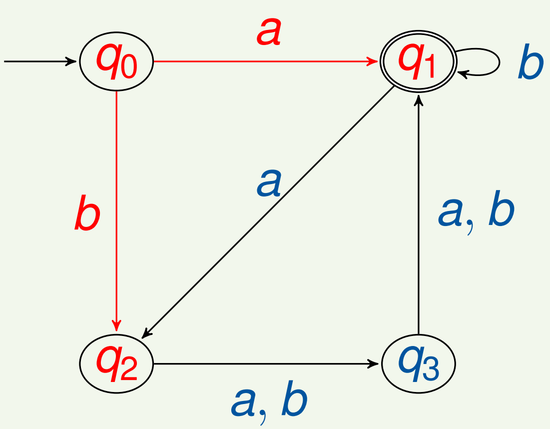


Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ with $N := Q$, $S := q_0$:

From Regular to Context-Free Languages

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:

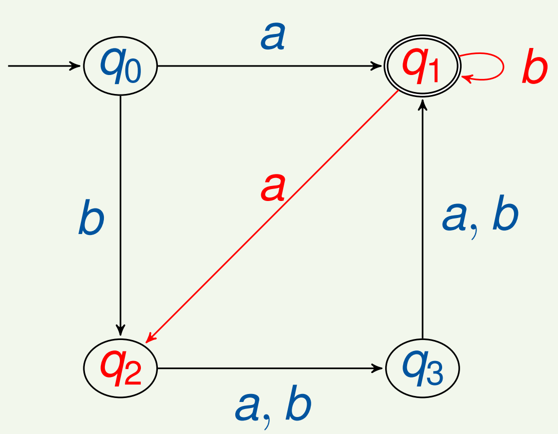


Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ with $N := Q$, $S := q_0$:

$$q_0 \rightarrow a\, q_1 \mid b\, q_2$$

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:

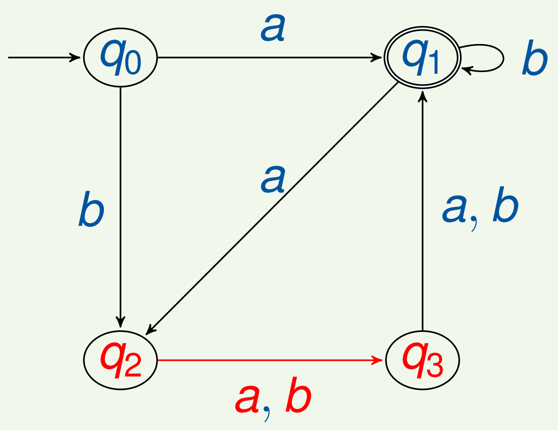


Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ with $N := Q$, $S := q_0$:

$$q_0 \rightarrow a\,q_1 \mid b\,q_2$$
$$q_1 \rightarrow a\,q_2 \mid b\,q_1 \mid \varepsilon$$

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:

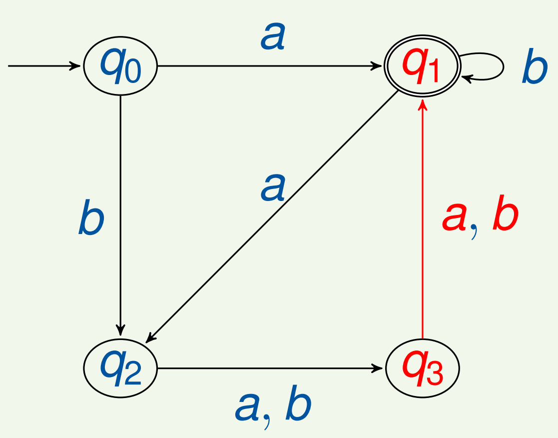


Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ with $N := Q$, $S := q_0$:

$$\begin{aligned} q_0 &\rightarrow a\, q_1 \mid b\, q_2 \\ q_1 &\rightarrow a\, q_2 \mid b\, q_1 \mid \varepsilon \\ q_2 &\rightarrow a\, q_3 \mid b\, q_3 \end{aligned}$$

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$:



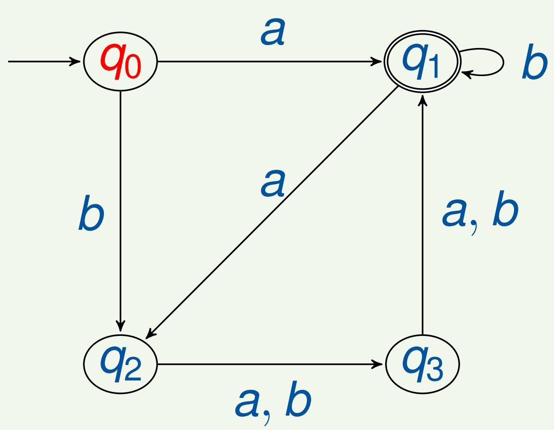
Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S\rangle$ with $N := Q$, $S := q_0$:

$$
\begin{aligned}
q_0 &\to a\,q_1 \mid b\,q_2\\
q_1 &\to a\,q_2 \mid b\,q_1 \mid \varepsilon\\
q_2 &\to a\,q_3 \mid b\,q_3\\
q_3 &\to a\,q_1 \mid b\,q_1
\end{aligned}
$$

From Regular to Context-Free Languages

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$:



Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S\rangle$ with $N := Q$, $S := q_0$:

$$\begin{aligned} q_0 &\to a\,q_1 \mid b\,q_2 \\ q_1 &\to a\,q_2 \mid b\,q_1 \mid \varepsilon \\ q_2 &\to a\,q_3 \mid b\,q_3 \\ q_3 &\to a\,q_1 \mid b\,q_1 \end{aligned}$$

E.g., $\mathfrak{A}$'s run on input $baab \in L(\mathfrak{A})$ is simulated by the following derivation in $G_{\mathfrak{A}}$:

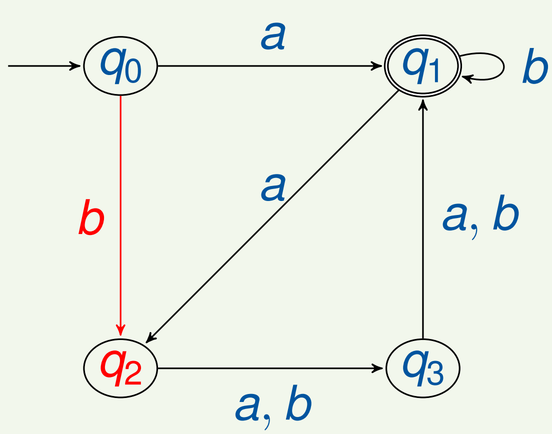
$$q_0$$

Foundations of Informatics/Formal Languages and Processes, Part B
Thomas Noll
b-it Bonn; 02–06 March 2020

From Regular to Context-Free Languages

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$:



Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S\rangle$ with $N := Q$, $S := q_0$:

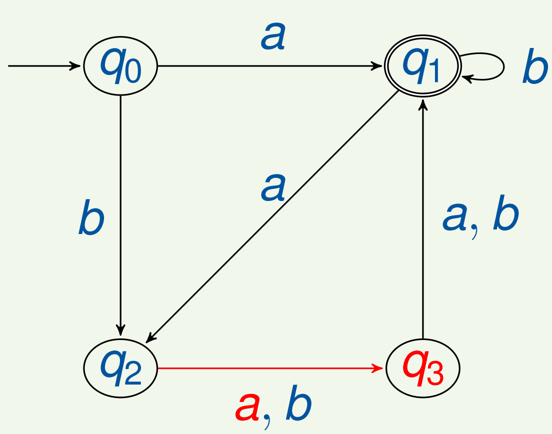
$$
\begin{aligned}
q_0 &\to a\,q_1 \mid b\,q_2 \\
q_1 &\to a\,q_2 \mid b\,q_1 \mid \varepsilon \\
q_2 &\to a\,q_3 \mid b\,q_3 \\
q_3 &\to a\,q_1 \mid b\,q_1
\end{aligned}
$$

E.g., $\mathfrak{A}$'s run on input $baab \in L(\mathfrak{A})$ is simulated by the following derivation in $G_{\mathfrak{A}}$:

$$q_0 \Rightarrow b\,q_2$$

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:



Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ with $N := Q$, $S := q_0$:

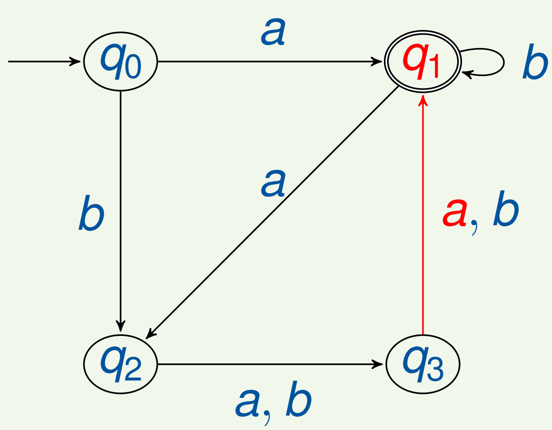
$$\begin{aligned} q_0 &\to a\,q_1 \mid b\,q_2 \\ q_1 &\to a\,q_2 \mid b\,q_1 \mid \varepsilon \\ q_2 &\to a\,q_3 \mid b\,q_3 \\ q_3 &\to a\,q_1 \mid b\,q_1 \end{aligned}$$

E.g., $\mathfrak{A}$'s run on input $baab \in L(\mathfrak{A})$ is simulated by the following derivation in $G_{\mathfrak{A}}$:

$$q_0 \Rightarrow b\,q_2 \Rightarrow b\,a\,q_3$$

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:



Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ with $N := Q$, $S := q_0$:

$$\begin{aligned} q_0 &\to a\,q_1 \mid b\,q_2 \\ q_1 &\to a\,q_2 \mid b\,q_1 \mid \varepsilon \\ q_2 &\to a\,q_3 \mid b\,q_3 \\ q_3 &\to a\,q_1 \mid b\,q_1 \end{aligned}$$

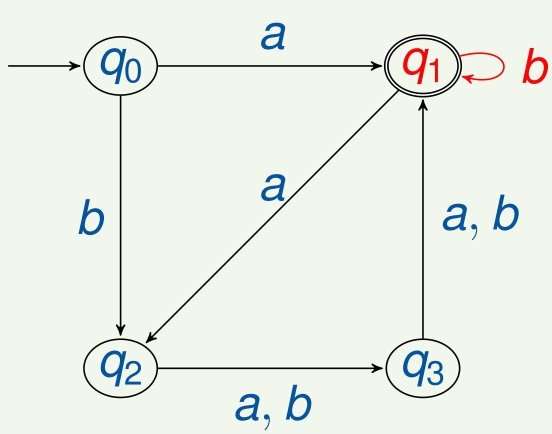
E.g., $\mathfrak{A}$'s run on input $baab \in L(\mathfrak{A})$ is simulated by the following derivation in $G_{\mathfrak{A}}$:

$$q_0 \Rightarrow b\,q_2 \Rightarrow b\,a\,q_3 \Rightarrow b\,a\,a\,q_1$$

From Regular to Context-Free Languages

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$:



Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S\rangle$ with $N := Q$, $S := q_0$:

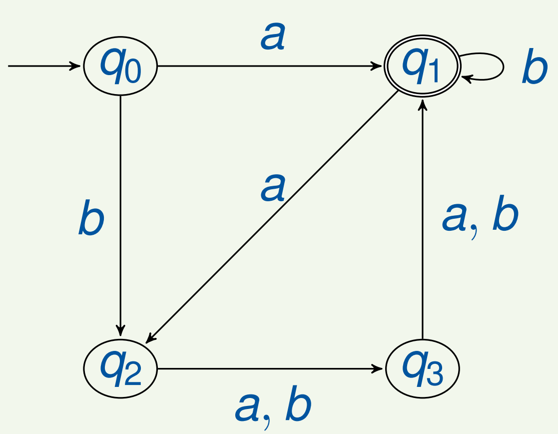
$$\begin{array}{lcl} q_0 & \to & a\,q_1 \mid b\,q_2 \\ q_1 & \to & a\,q_2 \mid b\,q_1 \mid \varepsilon \\ q_2 & \to & a\,q_3 \mid b\,q_3 \\ q_3 & \to & a\,q_1 \mid b\,q_1 \end{array}$$

E.g., $\mathfrak{A}$'s run on input $baab \in L(\mathfrak{A})$ is simulated by the following derivation in $G_{\mathfrak{A}}$:

$$q_0 \Rightarrow b\,q_2 \Rightarrow b\,a\,q_3 \Rightarrow b\,a\,a\,q_1 \Rightarrow b\,a\,a\,b\,q_1$$

Example B.7

DFA $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$:



Corresponding CFG $G_{\mathfrak{A}} := \langle N, \Sigma, P, S \rangle$ with $N := Q$, $S := q_0$:

$$
\begin{aligned}
q_0 &\to a\,q_1 \mid b\,q_2 \\
q_1 &\to a\,q_2 \mid b\,q_1 \mid \varepsilon \\
q_2 &\to a\,q_3 \mid b\,q_3 \\
q_3 &\to a\,q_1 \mid b\,q_1
\end{aligned}
$$

E.g., $\mathfrak{A}$'s run on input $baab \in L(\mathfrak{A})$ is simulated by the following derivation in $G_{\mathfrak{A}}$:

$$q_0 \Rightarrow b\,q_2 \Rightarrow b\,a\,q_3 \Rightarrow b\,a\,a\,q_1 \Rightarrow b\,a\,a\,b\,q_1 \Rightarrow b\,a\,a\,b$$

Summary: Context-Free vs. Regular Languages

Seen:

- CFLs are more expressive than regular languages

Summary: Context-Free vs. Regular Languages

Seen:

- CFLs are more expressive than regular languages

Next:

- Decidability of word problem

Outline of Part B

The Word Problem for CFL

Word Problem for CFL

Given CFG $G = \langle N, \Sigma, P, S \rangle$ and $w \in \Sigma^*$, decide whether $w \in L(G)$ or not.

The Word Problem for CFL

Word Problem for CFL

Given CFG $G = \langle N, \Sigma, P, S \rangle$ and $w \in \Sigma^*$, decide whether $w \in L(G)$ or not.

- Important problem with many applications
  - syntax analysis of programming languages
  - HTML parsers
  - ...
- For regular languages this was easy: just let the corresponding DFA run on w.
- But here: how to decide when to stop a derivation?
- **Solution:** establish normal form for grammars which guarantees that each nonterminal produces at least one terminal symbol

$\Rightarrow$ Only finitely many combinations to be inspected

Chomsky Normal Form

Definition B.8

A CFG is in Chomsky Normal Form (Chomsky NF) if every of its productions is of the form

$$A \to BC \quad \text{or} \quad A \to a$$

Chomsky Normal Form

Definition B.8

A CFG is in Chomsky Normal Form (Chomsky NF) if every of its productions is of the form

$$A \to BC \quad \text{or} \quad A \to a$$

Example B.9

Let $S \to ab \mid aSb$ be the grammar which generates $L := \{a^n b^n \mid n \geq 1\}$.
An equivalent grammar in Chomsky NF is

$$\begin{aligned}
S &\to AB \mid AC && \text{(generates } L\text{)}\\
A &\to a && \text{(generates } \{a\}\text{)}\\
B &\to b && \text{(generates } \{b\}\text{)}\\
C &\to SB && \text{(generates } \{a^n b^{n+1} \mid n \geq 1\}\text{)}
\end{aligned}$$

Conversion to Chomsky Normal Form

Theorem B.10

Every CFL L (with $\varepsilon \notin L$) can be generated by a CFG in Chomsky NF.

Conversion to Chomsky Normal Form

Theorem B.10

Every CFL L (with $\varepsilon \notin L$) can be generated by a CFG in Chomsky NF.

Proof.

Let L be a CFL, and let $G = \langle N, \Sigma, P, S \rangle$ be some CFG which generates L. The transformation of P into rules of the form $A \to BC$ and $A \to a$ proceeds in three steps:

1. terminal symbols only in rules of the form $A \to a$
   (thus all other rules have the shape $A \to A_1 \ldots A_n$)
2. elimination of "chain rules" of the form $A \to B$
3. elimination of rules of the form $A \to A_1 \ldots A_n$ where $n > 2$

(see following slides for details) □

Step 1: Only $A \rightarrow a$

Procedure

1. For every terminal symbol $a \in \Sigma$, introduce a new nonterminal symbol $B_a \in N$.
2. Add corresponding productions $B_a \rightarrow a$ to P.
3. In each original production $A \rightarrow \alpha$, replace every $a \in \Sigma$ with B_a.

This yields G'.

Step 1: Only $A \to a$

Procedure

1. For every terminal symbol $a \in \Sigma$, introduce a new nonterminal symbol $B_a \in N$.
2. Add corresponding productions $B_a \to a$ to P.
3. In each original production $A \to \alpha$, replace every $a \in \Sigma$ with B_a.

This yields G'.

Example B.11

$G: S \to ab \mid aSb$ is converted to

$$\begin{aligned} G': S &\to AB \mid ASB \\ A &\to a \\ B &\to b \end{aligned}$$

Step 2: Elimination of Chain Rules $A \to B$

Procedure

1. Determine all derivations $A_1 \Rightarrow \ldots \Rightarrow A_n$ with rules of the form $A \to B$ without repetition of nonterminals ($\Longrightarrow$ only finitely many!).
2. Determine all productions $A_n \to \alpha$ with $\alpha \notin N$.
3. Add corresponding productions $A_1 \to \alpha$ to P.
4. Remove all chain rules from P.

This yields G''.

Step 2: Elimination of Chain Rules $A \to B$

Procedure

1. Determine all derivations $A_1 \Rightarrow \ldots \Rightarrow A_n$ with rules of the form $A \to B$ without repetition of nonterminals ($\Longrightarrow$ only finitely many!).
2. Determine all productions $A_n \to \alpha$ with $\alpha \notin N$.
3. Add corresponding productions $A_1 \to \alpha$ to P.
4. Remove all chain rules from P.

This yields G''.

Example B.12

$$G' : \begin{array}{rcl} S & \to & A \\ A & \to & B \mid C \\ B & \to & A \mid DA \\ C & \to & c \\ D & \to & d \end{array}$$

is converted to

$$G'' : \begin{array}{rcl} S & \to & DA \mid c \\ A & \to & DA \mid c \\ B & \to & DA \mid c \\ C & \to & c \\ D & \to & d \end{array}$$

Step 3: Elimination of Rules $A \to A_1 \dots A_n$ with $n > 2$

Procedure

Iteratively apply the following transformation:

1. For every $A \to A_1 \dots A_n$ with $n > 2$, introduce a new nonterminal symbol $B \in N$.
2. Replace original production by $A \to A_1 B$.
3. Add new production $B \to A_2 \dots A_n$.

This yields G'''.

Step 3: Elimination of Rules $A \to A_1 \dots A_n$ with $n > 2$

Procedure

Iteratively apply the following transformation:

1. For every $A \to A_1 \dots A_n$ with $n > 2$, introduce a new nonterminal symbol $B \in N$.
2. Replace original production by $A \to A_1 B$.
3. Add new production $B \to A_2 \dots A_n$.

This yields G'''.

Example B.13

$$G'' : \begin{array}{lcl} S & \to & AB \mid ASB \\ A & \to & a \\ B & \to & b \end{array}$$

is converted to

$$G''' : \begin{array}{lcl} S & \to & AB \mid AC \\ A & \to & a \\ B & \to & b \\ C & \to & SB \end{array}$$

Summary: Chomsky Normal Form

Seen:

- Chomsky NF: all productions of the form $A \to BC$ or $A \to a$

Summary: Chomsky Normal Form

Seen:

- Chomsky NF: all productions of the form $A \to BC$ or $A \to a$

Next:

- Exploit Chomsky Normal Form to solve word problem for CFL

Outline of Part B

Word Problem for ε-free CFL

Given CFG $G = \langle N, \Sigma, P, S \rangle$ such that $\varepsilon \notin L(G)$ and $w \in \Sigma^+$, decide whether $w \in L(G)$ or not.

(If $w = \varepsilon$, then $w \in L(G)$ easily decidable for arbitrary G)

The Word Problem for CFL Revisited

Word Problem for ε-free CFL

Given CFG $G = \langle N, \Sigma, P, S \rangle$ such that $\varepsilon \notin L(G)$ and $w \in \Sigma^+$, decide whether $w \in L(G)$ or not.

(If $w = \varepsilon$, then $w \in L(G)$ easily decidable for arbitrary G)

Algorithm B.14 (by Cocke, Younger, Kasami – CYK algorithm)

1. *Transform G into Chomsky NF*
2. *Let $w = a_1 \ldots a_n$ ($n \geq 1$)*
3. *Let $w[i, j] := a_i \ldots a_j$ for every $1 \leq i \leq j \leq n$*
4. *Consider segments $w[i, j]$ in order of increasing length, starting with $w[i, i] = a_i$ (i.e., letters)*
5. *In each case, determine $N_{i,j} := \{A \in N \mid A \Rightarrow^* w[i, j]\}$ using a "dynamic programming" approach:*
   - $i = j$: $N_{i,i} = \{A \in N \mid A \to a_i \in P\}$
   - $i < j$: $N_{i,j} = \{A \in N \mid \exists B, C \in N, k \in \{i, \ldots, j-1\} : A \to BC \in P, B \in N_{i,k}, C \in N_{k+1,j}\}$
6. *Test whether $S \in N_{1,n}$ (and thus, whether $S \Rightarrow^* w[1, n] = w$)*

Matrix Representation of CYK Algorithm

	a_1	a_2	a_3	$\cdots$	a_n
$i \backslash j$	1	2	3	$\cdots$	n
1	$N_{1,1}$	$N_{1,2}$	$N_{1,3}$	$\cdots$	$N_{1,n}$
2	X	$N_{2,2}$	$N_{2,3}$	$\cdots$	$N_{2,n}$
3	X	X	$N_{3,3}$	$\cdots$	$N_{3,n}$
$\vdots$	$\vdots$	$\vdots$		$\cdots$	$\vdots$
n	X	X	$\cdots$	$\cdots$	$N_{n,n}$

Matrix Representation of CYK Algorithm

	a_1	a_2	a_3	$\cdots$	a_n
$i\backslash j$	1	2	3	$\cdots$	n
1	$N_{1,1}$	$N_{1,2}$	$N_{1,3}$	$\cdots$	$N_{1,n}$
2	X	$N_{2,2}$	$N_{2,3}$	$\cdots$	$N_{2,n}$
3	X	X	$N_{3,3}$	$\cdots$	$N_{3,n}$
$\vdots$	$\vdots$	$\vdots$		$\cdots$	$\vdots$
n	X	X	$\cdots$	$\cdots$	$N_{n,n}$

$$
\begin{aligned}
N_{1,1} &= \{A \in N \mid A \to a_1 \in P\} \\
N_{2,2} &= \{A \in N \mid A \to a_2 \in P\} \\
&\vdots
\end{aligned}
$$

Matrix Representation of CYK Algorithm

	a_1	a_2	a_3	$\cdots$	a_n
$i \backslash j$	1	2	3	$\cdots$	n
1	$N_{1,1}$	$N_{1,2}$	$N_{1,3}$	$\cdots$	$N_{1,n}$
2	X	$N_{2,2}$	$N_{2,3}$	$\cdots$	$N_{2,n}$
3	X	X	$N_{3,3}$	$\cdots$	$N_{3,n}$
$\vdots$	$\vdots$	$\vdots$		$\cdots$	$\vdots$
n	X	X	$\cdots$	$\cdots$	$N_{n,n}$

$$
\begin{aligned}
N_{1,1} &= \{A \in N \mid A \to a_1 \in P\} \\
N_{2,2} &= \{A \in N \mid A \to a_2 \in P\} \\
&\vdots \\
N_{1,2} &= \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{1,1}, C \in N_{2,2}\} \\
N_{2,3} &= \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{2,2}, C \in N_{3,3}\} \\
&\vdots
\end{aligned}
$$

Matrix Representation of CYK Algorithm

	a_1	a_2	a_3	$\cdots$	a_n
$i \backslash j$	1	2	3	$\cdots$	n
1	$N_{1,1}$	$N_{1,2}$	$N_{1,3}$	$\cdots$	$N_{1,n}$
2	X	$N_{2,2}$	$N_{2,3}$	$\cdots$	$N_{2,n}$
3	X	X	$N_{3,3}$	$\cdots$	$N_{3,n}$
$\vdots$	$\vdots$	$\vdots$		$\cdots$	$\vdots$
n	X	X	$\cdots$	$\cdots$	$N_{n,n}$

$$
\begin{aligned}
N_{1,1} &= \{A \in N \mid A \to a_1 \in P\}\\
N_{2,2} &= \{A \in N \mid A \to a_2 \in P\}\\
&\vdots\\
N_{1,2} &= \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{1,1}, C \in N_{2,2}\}\\
N_{2,3} &= \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{2,2}, C \in N_{3,3}\}\\
&\vdots\\
N_{1,3} &= \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{1,1}, C \in N_{2,3}\}\\
&\cup \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{1,2}, C \in N_{3,3}\}\\
N_{2,4} &= \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{2,2}, C \in N_{3,4}\}\\
&\cup \{A \in N \mid \exists B, C \in N : A \to BC \in P, B \in N_{2,3}, C \in N_{4,4}\}\\
&\vdots
\end{aligned}
$$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

Applying the CYK Algorithm

Example B.15

- G :
  $S \rightarrow SA \mid a$
  $A \rightarrow BS$
  $B \rightarrow BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1						
2	X					
3	X	X				
4	X	X	X			
5	X	X	X	X		
6	X	X	X	X	X	

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$					
2	X					
3	X	X	$\{S\}$			
4	X	X	X	$\{S\}$		
5	X	X	X	X		
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$					
2	X	$\{B\}$				
3	X	X	$\{S\}$			
4	X	X	X	$\{S\}$		
5	X	X	X	X	$\{B\}$	
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$				
2	X	$\{B\}$				
3	X	X	$\{S\}$	$\emptyset$		
4	X	X	X	$\{S\}$	$\emptyset$	
5	X	X	X	X	$\{B\}$	
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G:
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$				
2	X	$\{B\}$	$\{A\quad\}$			
3	X	X	$\{S\}$	$\emptyset$		
4	X	X	X	$\{S\}$	$\emptyset$	
5	X	X	X	X	$\{B\}$	$\{A\quad\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i\backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$				
2	X	$\{B\}$	$\{A, B\}$			
3	X	X	$\{S\}$	$\emptyset$		
4	X	X	X	$\{S\}$	$\emptyset$	
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$			
2	X	$\{B\}$	$\{A, B\}$			
3	X	X	$\{S\}$	$\emptyset$		
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$			
2	X	$\{B\}$	$\{A, B\}$	$\{A \quad\}$		
3	X	X	$\{S\}$	$\emptyset$		
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$			
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$		
3	X	X	$\{S\}$	$\emptyset$		
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$			
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$		
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$		
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$		
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$		
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$	$\{B\}$	
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$		
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$	$\{B\}$	
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$	$\{B\}$	
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$	$\{B\}$	$\{A \quad\}$
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$	$\{B\}$	$\{A, B\}$
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	$\{S\}$
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$	$\{B\}$	$\{A, B\}$
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

Applying the CYK Algorithm

Example B.15

- G :
  $S \to SA \mid a$
  $A \to BS$
  $B \to BB \mid BS \mid b \mid c$
- $w = abaaba$

	a	b	a	a	b	a
$i \backslash j$	1	2	3	4	5	6
1	$\{S\}$	$\emptyset$	$\{S\}$	$\{S\}$	$\emptyset$	$\{S\}$
2	X	$\{B\}$	$\{A, B\}$	$\{A, B\}$	$\{B\}$	$\{A, B\}$
3	X	X	$\{S\}$	$\emptyset$	$\emptyset$	$\emptyset$
4	X	X	X	$\{S\}$	$\emptyset$	$\{S\}$
5	X	X	X	X	$\{B\}$	$\{A, B\}$
6	X	X	X	X	X	$\{S\}$

$$S \in N_{1,6} \implies w = abaaba \in L(G)$$

Summary: The Word Problem for Context-Free Languages

Seen:

- Given CFG G and $w \in \Sigma^*$, decide whether $w \in L(G)$ or not
- Decidable using CYK algorithm (based on dynamic programming)
- Cubic complexity

Summary: The Word Problem for Context-Free Languages

Seen:

- Given CFG G and $w \in \Sigma^*$, decide whether $w \in L(G)$ or not
- Decidable using CYK algorithm (based on dynamic programming)
- Cubic complexity

Next:

- Emptiness problem

Outline of Part B

The Emptiness Problem

Emptiness Problem for CFL

Given CFG $G = \langle N, \Sigma, P, S \rangle$, decide whether $L(G) = \emptyset$ or not.

The Emptiness Problem

Emptiness Problem for CFL

Given CFG $G = \langle N, \Sigma, P, S \rangle$, decide whether $L(G) = \emptyset$ or not.

- Important problem with many applications
  - consistency of context-free language definitions
  - correctness properties of recursive programs
  - ...
- For regular languages this was easy: check in the corresponding DFA whether some final state is reachable from the initial state.
- Here: test whether start symbol is productive, i.e., whether it generates a terminal word

The Emptiness Test

Algorithm B.16 (Emptiness Test)

Input: $G = \langle N, \Sigma, P, S \rangle$

Question: $L(G) = \emptyset$?

Procedure: *mark every* $a \in \Sigma$ *as productive*;

```
repeat
   if there is A → α ∈ P such that all symbols in α productive then
      mark A as productive;
   end;
until no further productive symbols found;
```

Output: *"no" if S productive, otherwise "yes"*

The Emptiness Test

Algorithm B.16 (Emptiness Test)

Input: $G = \langle N, \Sigma, P, S \rangle$

Question: $L(G) = \emptyset$?

Procedure: *mark every* $a \in \Sigma$ *as productive*;

```
repeat
   if there is A → α ∈ P such that all symbols in α productive then
      mark A as productive;
   end;
until no further productive symbols found;
```

Output: *"no" if S productive, otherwise "yes"*

Example B.17

$$
\begin{aligned}
G:\quad S &\to AB \mid CA \\
A &\to a \\
B &\to BC \mid AB \\
C &\to aB \mid b
\end{aligned}
$$

The Emptiness Test

Algorithm B.16 (Emptiness Test)

Input: $G = \langle N, \Sigma, P, S \rangle$

Question: $L(G) = \emptyset$?

Procedure: *mark every* $a \in \Sigma$ *as productive*;

```
repeat
  if there is A → α ∈ P such that all symbols in α productive then
    mark A as productive;
  end;
until no further productive symbols found;
```

Output: "no" if S productive, otherwise "yes"

Example B.17

$$
\begin{aligned}
G:\quad S &\to AB \mid CA\\
A &\to a\\
B &\to BC \mid AB\\
C &\to aB \mid b
\end{aligned}
$$

1. Initalisation

Algorithm B.16 (Emptiness Test)

Input: $G = \langle N, \Sigma, P, S \rangle$

Question: $L(G) = \emptyset$?

Procedure: *mark every* $a \in \Sigma$ *as productive*;

```
repeat
  if there is A → α ∈ P such that all symbols in α productive then
    mark A as productive;
  end;
until no further productive symbols found;
```

Output: *"no" if S productive, otherwise "yes"*

Example B.17

$$
\begin{aligned}
G:\quad S &\to AB \mid CA \\
A &\to a \\
B &\to BC \mid AB \\
C &\to aB \mid b
\end{aligned}
$$

1. Initalisation
2. 1st iteration

The Emptiness Test

Algorithm B.16 (Emptiness Test)

Input: $G = \langle N, \Sigma, P, S \rangle$

Question: $L(G) = \emptyset$?

Procedure: mark every $a \in \Sigma$ as productive;

```
repeat
   if there is A → α ∈ P such that all symbols in α productive then
      mark A as productive;
   end;
until no further productive symbols found;
```

Output: "no" if S productive, otherwise "yes"

Example B.17

$$
\begin{aligned}
G:\ & S \to AB \mid CA \\
& A \to a \\
& B \to BC \mid AB \\
& C \to aB \mid b
\end{aligned}
$$

1. Initalisation
2. 1st iteration
3. 2nd iteration

The Emptiness Test

Algorithm B.16 (Emptiness Test)

Input: $G = \langle N, \Sigma, P, S \rangle$

Question: $L(G) = \emptyset$?

Procedure: mark every $a \in \Sigma$ as productive;

```
repeat
   if there is A → α ∈ P such that all symbols in α productive then
      mark A as productive;
   end;
until no further productive symbols found;
```

Output: "no" if S productive, otherwise "yes"

Example B.17

$$\begin{aligned} G:\quad S &\to AB \mid CA \\ A &\to a \\ B &\to BC \mid AB \\ C &\to aB \mid b \end{aligned}$$

1. Initalisation
2. 1st iteration
3. 2nd iteration

S productive $\Longrightarrow L(G) \neq \emptyset$

Summary: The Emptiness Problem for CFLs

Seen:

- Emptiness problem decidable based on productivity of symbols

Summary: The Emptiness Problem for CFLs

Seen:

- Emptiness problem decidable based on productivity of symbols

Next:

- Closure properties of CFLs

Outline of Part B

Context-Free Grammars and Languages

Context-Free vs. Regular Languages

Chomsky Normal Form

The Word Problem for Context-Free Languages

The Emptiness Problem for CFLs

Closure Properties of CFLs

Pushdown Automata

Outlook

Positive Results

Theorem B.18

The set of CFLs is closed under concatenation, union, and iteration.

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Positive Results

Theorem B.18

The set of CFLs is closed under concatenation, union, and iteration.

Proof.

For $i = 1, 2$, let $G_i = \langle N_i, \Sigma, P_i, S_i \rangle$ with $L_i := L(G_i)$ and $N_1 \cap N_2 = \emptyset$. Then

Positive Results

Theorem B.18

The set of CFLs is closed under concatenation, union, and iteration.

Proof.

For $i = 1, 2$, let $G_i = \langle N_i, \Sigma, P_i, S_i \rangle$ with $L_i := L(G_i)$ and $N_1 \cap N_2 = \emptyset$. Then

- $G := \langle N, \Sigma, P, S \rangle$ with $N := \{S\} \cup N_1 \cup N_2$ and $P := \{S \to S_1 S_2\} \cup P_1 \cup P_2$ generates $L_1 \cdot L_2$;

Positive Results

Theorem B.18

The set of CFLs is closed under concatenation, union, and iteration.

Proof.

For $i = 1, 2$, let $G_i = \langle N_i, \Sigma, P_i, S_i \rangle$ with $L_i := L(G_i)$ and $N_1 \cap N_2 = \emptyset$. Then

- $G := \langle N, \Sigma, P, S \rangle$ with $N := \{S\} \cup N_1 \cup N_2$ and $P := \{S \to S_1 S_2\} \cup P_1 \cup P_2$ generates $L_1 \cdot L_2$;
- $G := \langle N, \Sigma, P, S \rangle$ with $N := \{S\} \cup N_1 \cup N_2$ and $P := \{S \to S_1 \mid S_2\} \cup P_1 \cup P_2$ generates $L_1 \cup L_2$; and

Positive Results

Theorem B.18

The set of CFLs is closed under concatenation, union, and iteration.

Proof.

For $i = 1, 2$, let $G_i = \langle N_i, \Sigma, P_i, S_i \rangle$ with $L_i := L(G_i)$ and $N_1 \cap N_2 = \emptyset$. Then

- $G := \langle N, \Sigma, P, S \rangle$ with $N := \{S\} \cup N_1 \cup N_2$ and $P := \{S \to S_1 S_2\} \cup P_1 \cup P_2$ generates $L_1 \cdot L_2$;
- $G := \langle N, \Sigma, P, S \rangle$ with $N := \{S\} \cup N_1 \cup N_2$ and $P := \{S \to S_1 \mid S_2\} \cup P_1 \cup P_2$ generates $L_1 \cup L_2$; and
- $G := \langle N, \Sigma, P, S \rangle$ with $N := \{S\} \cup N_1$ and $P := \{S \to \varepsilon \mid S_1 S\} \cup P_1$ generates L_1^*.

$\square$

Negative Results

Theorem B.19

The set of CFLs is not closed under intersection and complement.

Negative Results

Theorem B.19

The set of CFLs is not closed under intersection and complement.

Proof.

- Intersection: both $L_1 := \{a^k b^k c^l \mid k, l \in \mathbb{N}\}$ and $L_2 := \{a^k b^l c^l \mid k, l \in \mathbb{N}\}$ are CFLs, but not $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ (without proof).

Negative Results

Theorem B.19

The set of CFLs is not closed under intersection and complement.

Proof.

- Intersection: both $L_1 := \{a^k b^k c^l \mid k, l \in \mathbb{N}\}$ and $L_2 := \{a^k b^l c^l \mid k, l \in \mathbb{N}\}$ are CFLs, but not $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ (without proof).
- Complement: if CFLs were closed under complement, then also under intersection (as $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$).

□

Overview of Decidability and Closure Results

Decidability Results			
Class	$w \in L$	$L = \emptyset$	$L_1 = L_2$
Reg	+ (A.37)	+ (A.39)	+ (A.41)
CFL	+ (B.14)	+ (B.16)	–

Overview of Decidability and Closure Results

Decidability Results			
Class	$w \in L$	$L = \emptyset$	$L_1 = L_2$
Reg	+ (A.37)	+ (A.39)	+ (A.41)
CFL	+ (B.14)	+ (B.16)	–

Closure Results					
Class	$L_1 \cdot L_2$	$L_1 \cup L_2$	$L_1 \cap L_2$	$\overline{L}$	L^*
Reg	+ (A.28)	+ (A.18)	+ (A.16)	+ (A.14)	+ (A.29)
CFL	+ (B.18)	+ (B.18)	– (B.19)	– (B.19)	+ (B.18)

Closure Properties

Seen:

- Closure under concatenation, union and iteration
- Non-closure under intersection and complement

Closure Properties

Seen:

- Closure under concatenation, union and iteration
- Non-closure under intersection and complement

Next:

- Automata model for CFLs

Outline of Part B

Pushdown Automata I

- **Goal:** introduce an automata model which exactly accepts CFLs
- **Clear:** DFA not sufficient
  (missing "counting capability", e.g. for $\{a^n b^n \mid n \geq 1\}$)
- DFA will be extended to pushdown automata by
  - adding a pushdown store which stores symbols from a pushdown alphabet and uses a special bottom symbol
  - adding push and pop operations to transitions

Definition B.20

A pushdown automaton (PDA) is of the form $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F \rangle$ where

- Q is a finite set of states
- Σ is the (finite) input alphabet
- Γ is the (finite) pushdown alphabet
- $\Delta \subseteq (Q \times \Gamma \times \Sigma_\varepsilon) \times (Q \times \Gamma^*)$ is a finite set of transitions
- $q_0 \in Q$ is the initial state
- Z_0 is the (pushdown) bottom symbol
- $F \subseteq Q$ is a set of final states

Interpretation of $((q, Z, x), (q', \delta)) \in \Delta$: if the PDA $\mathfrak{A}$ is in state q where Z is on top of the stack and x is the next input symbol (or empty), then $\mathfrak{A}$ reads x, replaces Z by δ, and changes into the state q'.

Configurations, Runs, Acceptance

Definition B.21

Let $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F \rangle$ be a PDA.

- An element of $Q \times \Gamma^* \times \Sigma^*$ is called a configuration of $\mathfrak{A}$.
- The initial configuration for input $w \in \Sigma^*$ is given by (q_0, Z_0, w).
- The set of final configurations is given by $F \times \{\varepsilon\} \times \{\varepsilon\}$.
- If $((q, Z, x), (q', \delta)) \in \Delta$, then $(q, Z\gamma, xw) \vdash (q', \delta\gamma, w)$ for every $\gamma \in \Gamma^*$, $w \in \Sigma^*$.

Configurations, Runs, Acceptance

Definition B.21

Let $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F \rangle$ be a PDA.

- An element of $Q \times \Gamma^* \times \Sigma^*$ is called a configuration of $\mathfrak{A}$.
- The initial configuration for input $w \in \Sigma^*$ is given by (q_0, Z_0, w).
- The set of final configurations is given by $F \times \{\varepsilon\} \times \{\varepsilon\}$.
- If $((q, Z, x), (q', \delta)) \in \Delta$, then $(q, Z\gamma, xw) \vdash (q', \delta\gamma, w)$ for every $\gamma \in \Gamma^*$, $w \in \Sigma^*$.
- $\mathfrak{A}$ accepts $w \in \Sigma^*$ if $(q_0, Z_0, w) \vdash^* (q, \varepsilon, \varepsilon)$ for some $q \in F$.
- The language accepted by $\mathfrak{A}$ is $L(\mathfrak{A}) := \{w \in \Sigma^* \mid \mathfrak{A} \text{ accepts } w\}$.
- A language L is called PDA-recognisable if $L = L(\mathfrak{A})$ for some PDA $\mathfrak{A}$.
- Two PDA $\mathfrak{A}_1, \mathfrak{A}_2$ are called equivalent if $L(\mathfrak{A}_1) = L(\mathfrak{A}_2)$.

Examples

Example B.22

1. PDA which recognises $L = \{a^n b^n \mid n \geq 1\}$
   (on the board)

Examples

Example B.22

1. PDA which recognises $L = \{a^n b^n \mid n \geq 1\}$
   (on the board)
2. PDA which recognises $L = \{ww^R \mid w \in \{a, b\}^*\}$
   (palindromes of even length; on the board)

Examples

Example B.22

1. PDA which recognises $L = \{a^n b^n \mid n \geq 1\}$
   (on the board)
2. PDA which recognises $L = \{ww^R \mid w \in \{a, b\}^*\}$
   (palindromes of even length; on the board)

Observation: $\mathfrak{A}_2$ is nondeterministic: whenever a construction transition is applicable, the pushdown could also be deconstructed

Deterministic PDA

Definition B.23

A PDA $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F \rangle$ is called deterministic (DPDA) if for every $q \in Q, Z \in \Gamma$,

1. for every $x \in \Sigma_\varepsilon$, there is at most one (q, Z, x)-transition in Δ and
2. if there is a (q, Z, a)-transition in Δ for some $a \in \Sigma$, then there is no (q, Z, ε)-transition in Δ.

Remark: this excludes two types of nondeterminism:

1. if $((q, Z, x), (q'_1, \delta_1)), ((q, Z, x), (q'_2, \delta_2)) \in \Delta$:
$$(q'_1, \delta_1\gamma, w) \dashv (q, Z\gamma, xw) \vdash (q'_2, \delta_2\gamma, w)$$
2. if $((q, Z, a), (q'_1, \delta_1)), ((q, Z, \varepsilon), (q'_2, \delta_2)) \in \Delta$:
$$(q'_1, \delta_1\gamma, w) \dashv (q, Z\gamma, aw) \vdash (q'_2, \delta_2\gamma, aw)$$

Deterministic PDA

Definition B.23

A PDA $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F \rangle$ is called deterministic (DPDA) if for every $q \in Q, Z \in \Gamma$,

1. for every $x \in \Sigma_\varepsilon$, there is at most one (q, Z, x)-transition in Δ and
2. if there is a (q, Z, a)-transition in Δ for some $a \in \Sigma$, then there is no (q, Z, ε)-transition in Δ.

Remark: this excludes two types of nondeterminism:

1. if $((q, Z, x), (q_1', \delta_1)), ((q, Z, x), (q_2', \delta_2)) \in \Delta$:
$$(q_1', \delta_1\gamma, w) \dashv (q, Z\gamma, xw) \vdash (q_2', \delta_2\gamma, w)$$
2. if $((q, Z, a), (q_1', \delta_1)), ((q, Z, \varepsilon), (q_2', \delta_2)) \in \Delta$:
$$(q_1', \delta_1\gamma, w) \dashv (q, Z\gamma, aw) \vdash (q_2', \delta_2\gamma, aw)$$

Corollary B.24

In a DPDA, every configuration has at most one $\vdash$-successor.

Expressiveness of DPDA

One can show: determinism restricts the set of acceptable languages (DPDA-recognisable languages are closed under complement, which is generally not true for PDA-recognisable languages)

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Expressiveness of DPDA

One can show: determinism restricts the set of acceptable languages (DPDA-recognisable languages are closed under complement, which is generally not true for PDA-recognisable languages)

Example B.25

The set of palindromes of even length is PDA-recognisable, but not DPDA-recognisable (without proof).

PDA and Context-Free Languages I

Theorem B.26

A language is context-free iff it is PDA-recognisable.

PDA and Context-Free Languages I

Theorem B.26

A language is context-free iff it is PDA-recognisable.

Proof.

$\Leftarrow$: omitted

$\Rightarrow$: let $G = \langle N, \Sigma, P, S \rangle$ be a CFG. Construction of PDA $\mathfrak{A}_G$ recognising $L(G)$:

- $\mathfrak{A}_G$ simulates a derivation of G where always the leftmost nonterminal of a sentence is replaced ("leftmost derivation")
- begin with S on pushdown
- if nonterminal on top: apply a corresponding production rule
- if terminal on top: match with next input symbol

(cf. formal construction on following slide)

□

PDA and Context-Free Languages II

Proof of Theorem B.26 (continued).

$\Rightarrow$: Formally: $\mathfrak{A}_G := \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F \rangle$ is given by

- $Q := \{q_0\}$
- $\Gamma := N \cup \Sigma$
- for each $A \to \alpha \in P$: $((q_0, A, \varepsilon), (q_0, \alpha)) \in \Delta$
- for each $a \in \Sigma$: $((q_0, a, a), (q_0, \varepsilon)) \in \Delta$
- $Z_0 := S$
- $F := Q$

□

PDA and Context-Free Languages II

Proof of Theorem B.26 (continued).

$\Rightarrow$: Formally: $\mathfrak{A}_G := \langle Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F\rangle$ is given by

- $Q := \{q_0\}$
- $\Gamma := N \cup \Sigma$
- for each $A \to \alpha \in P$: $((q_0, A, \varepsilon), (q_0, \alpha)) \in \Delta$
- for each $a \in \Sigma$: $((q_0, a, a), (q_0, \varepsilon)) \in \Delta$
- $Z_0 := S$
- $F := Q$

$\square$

Example B.27

"Bracket language", given by G:

$$S \to \langle\rangle \mid \langle S\rangle \mid SS$$

(on the board)

Outline of Part B

Outlook

- Equivalence problem for CFG and PDA ("$L(X_1) = L(X_2)$?")
  (generally undecidable, decidable for DPDA)
- Pumping Lemma for CFL
- Greibach Normal Form for CFG
- Construction of parsers for compilers
- Non-context-free grammars and languages (context-sensitive and recursively enumerable languages, Turing machines—see Week 4)