Foundations of Informatics: a Bridging Course

Week 3: Formal Languages and Processes
Part B: Context-Free Languages
b-it Bonn; March 12–16, 2018

Erika Ábrahám
Theory of Hybrid Systems Group
RWTH Aachen University

Thanks to Thomas Noll for providing slides

https://ths.rwth-aachen.de/teaching/ws18/b-it-bridging-course/

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.2 (continued)

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

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

Introductory Example II

Example B.2 (continued)

Here we abbreviate $\langle Expression \rangle$ as E, and use "$\longrightarrow$" 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:

$$E \;\Rightarrow\; E * E$$

Introductory Example II

Example B.2 (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:

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

Introductory Example II

Example B.2 (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:

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

Introductory Example II

Example B.2 (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:

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

Introductory Example II

Example B.2 (continued)

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

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

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.2 (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:

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

Context-Free Grammars I

Definition B.3

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.4

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.4

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.5

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) 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 $1 \leq i \leq n$ (notation: $\beta \Rightarrow^* \gamma$).

A word $w \in \Sigma^*$ is called **derivable** in G if $S \Rightarrow^* w$.

Context-Free Languages I

Definition B.5

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) 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 $1 \leq i \leq n$ (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.6

The language $\{a^n b^n \mid n \geq 1\}$ is context-free. It is generated by the grammar $G = \langle N, \Sigma, P, S \rangle$ with

$$N = \{S\}$$

$$\Sigma = \{a, b\}$$

$$P = \{S \rightarrow aSb \mid ab\}$$

(proof: generating $a^n b^n$ requires exactly $n - 1$ applications of the first and one concluding application of the second rule)

Context-Free Languages II

Example B.6

The language $\{a^n b^n \mid n \geq 1\}$ is context-free. It is generated by the grammar $G = \langle N, \Sigma, P, S \rangle$ with

$N = \{S\}$

$\Sigma = \{a, b\}$

$P = \{S \to aSb \mid ab\}$

(proof: generating $a^n b^n$ requires exactly $n - 1$ applications of the first and one concluding application of the second rule)

Remark: illustration of derivations by **derivation trees**

- root labelled by start symbol
- leafs labelled by terminal symbols
- successors of node labelled according to right-hand side of production rule

(example on the board)

Context-Free Grammars and Languages

Seen:

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

Context-Free Grammars and Languages

Seen:

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

Open:

- Relation between context-free and regular languages

Outline of Part B

Context-Free vs. Regular Languages

Theorem B.7

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

(In other words: the class of regular languages is a **proper subset** of CFLs.)

Context-Free vs. Regular Languages

Theorem B.7

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

(In other words: the class of regular languages is 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.
   $G := \langle N, \Sigma, P, S \rangle$ 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, and vice versa. Thus $L(\mathfrak{A}) = L(G)$ (example on the board).
2. An example is $\{a^n b^n \mid n \geq 1\}$ (see Ex. B.6). □

Context-Free vs. Regular Languages

Context-Free Grammars and Languages

Seen:

CFLs are more expressive than regular languages

Context-Free vs. Regular Languages

Context-Free Grammars and Languages

Seen:

CFLs are more expressive than regular languages

Open:

Decidability of word problem

Outline of Part B

The Word Problem for CFLs

The Word Problem

Goal: given $G = \langle N, \Sigma, P, S \rangle$ and $w \in \Sigma^*$, decide whether $w \in L(G)$ or not

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 I

Definition B.8

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

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

Chomsky Normal Form I

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

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

Chomsky Normal Form II

Theorem B.10

Every CFL L (with $\varepsilon \notin L$) is generatable by a CFG in Chomsky NF.

Chomsky Normal Form II

Theorem B.10

Every CFL L (with $\varepsilon \notin L$) is generatable 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 $A \rightarrow BC$ and $A \rightarrow a$ proceeds in three steps:

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

(details omitted) □

The Word Problem Revisited

Goal: given $w \in \Sigma^+$ and $G = \langle N, \Sigma, P, S \rangle$ such that $\varepsilon \notin L(G)$, decide if $w \in L(G)$ or not

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

Approach by Cocke, Younger, Kasami (**CYK algorithm**):

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

The CYK Algorithm I

Algorithm B.11 (CYK Algorithm)

Input: $G = \langle N, \Sigma, P, S\rangle$ *in Chomsky NF,* $w = a_1 \dots a_n \in \Sigma^+$

Question: $w \in L(G)$*?*

Procedure:
```
for i := 1 to n do
    N_{i,i} := {A ∈ N | A → a_i ∈ P}
next i;
for d := 1 to n − 1 do    % compute N_{i,i+d}
    for i := 1 to n − d do
        j := i + d; N_{i,j} := ∅;
        for k := i to j − 1 do
            N_{i,j} := N_{i,j} ∪ {A ∈ N | ∃A → BC ∈ P : B ∈ N_{i,k}, C ∈ N_{k+1,j}}
        next k
    next i
next d
```

Output: *"yes" if* $S \in N_{1,n}$*, otherwise "no"*

The CYK Algorithm II

Example B.12

$$
\begin{aligned}
G:\ & S \to SA \mid a \\
& A \to BS \\
& B \to BB \mid BS \mid b \mid c
\end{aligned}
$$

$w = abaaba$

Matrix representation of $N_{i,j}$

(on the board)

The Word Problem for Context-Free Languages

Seen:

Word problem decidable using CYK algorithm

The Word Problem for Context-Free Languages

Seen:

Word problem decidable using CYK algorithm

Open:

Emptiness problem

Outline of Part B

The Emptiness Problem for CFLs

The Emptiness Problem

Goal: given $G = \langle N, \Sigma, P, S \rangle$, decide whether $L(G) = \emptyset$ or not

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.13 (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.13 (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.14

$G: \; S \to AB \mid CA \qquad A \to a$ (on the board)

$\quad\;\; B \to BC \mid AB \qquad C \to aB \mid b$

The Emptiness Problem for CFLs

Seen:

Emptiness problem decidable based on productivity of symbols

The Emptiness Problem for CFLs

Seen:

Emptiness problem decidable based on productivity of symbols

Open:

Closure properties of CFLs

Outline of Part B

Closure Properties of CFLs

Positive Results

Theorem B.15

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

Positive Results

Theorem B.15

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.15

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.15

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.15

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^*.

□

Negative Results

Theorem B.16

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

Negative Results

Theorem B.16

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

Proof.

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.16

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

Proof.

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).

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.38)	+ (A.40)	+ (A.42)
CFL	+ (B.11)	+ (B.13)	–

Overview of Decidability and Closure Results

Decidability Results			
Class	$w \in L$	$L = \emptyset$	$L_1 = L_2$
Reg	+ (A.38)	+ (A.40)	+ (A.42)
CFL	+ (B.11)	+ (B.13)	–

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.15)	+ (B.15)	– (B.16)	– (B.16)	+ (B.15)

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

Open:

- 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

Pushdown Automata II

Definition B.17

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.18

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.18

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.19

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

Examples

Example B.19

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.19

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.20

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.20

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.21

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)

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.22

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

PDA and Context-Free Languages I

Theorem B.23

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

PDA and Context-Free Languages I

Theorem B.23

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.23 (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.23 (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 $a \in \Sigma$: $((q_0, a, a), (q_0, \varepsilon)) \in \Delta$
$Z_0 := S$
$F := Q$

□

Example B.24

"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)