

Foundations of Informatics: a Bridging Course

Week 3: Formal Languages and Semantics

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

<http://cosec.bit.uni-bonn.de/students/teaching/09us/09us-bridgingcourse/>
<http://www-i2.informatik.rwth-aachen.de/i2/b-it09/>

B-IT, Bonn, Winter semester 2009/10

Part II

Context-Free Languages

- 1 Context-Free Grammars and Languages
- 2 Context-Free and Regular Languages
- 3 The Word Problem for Context-Free Languages
- 4 The Emptiness Problem for CFLs
- 5 Pushdown Automata
- 6 Closure Properties of CFLs
- 7 Outlook

Introductory Example I

Example II.1

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

Here: **simple arithmetic expressions**

$$\begin{aligned}\langle \text{Expression} \rangle &::= 0 \\ &\quad | \quad 1 \\ &\quad | \quad \langle \text{Expression} \rangle + \langle \text{Expression} \rangle \\ &\quad | \quad \langle \text{Expression} \rangle * \langle \text{Expression} \rangle \\ &\quad | \quad (\langle \text{Expression} \rangle)\end{aligned}$$

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 II.2 (continued)

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

Thus:

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

Introductory Example II

Example II.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 **applying rules** to the start symbol E :

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

Definition II.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 $A \rightarrow \alpha$ where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
- $S \in N$ is a **start symbol**

Example II.4

For the above example, we have:

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

Example II.4

For the above example, we have:

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

Naming conventions:

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

⇒ grammar completely defined by productions

Definition II.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 $\pi = A \rightarrow \alpha \in P$ 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: $\beta \xrightarrow{\pi} \gamma$ or just $\beta \Rightarrow \gamma$).
- A **derivation** (of length n) of γ from β is a sequence of direct derivations of the form $\delta_0 \Rightarrow \delta_1 \Rightarrow \dots \Rightarrow \delta_n$ 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 $L(G_1) = L(G_2)$.

Example II.6

The language $\{a^n b^n \mid n \geq 1\}$ is context-free (but not regular—see Ex. I.51). 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: on the board)

Example II.6

The language $\{a^n b^n \mid n \geq 1\}$ is context-free (but not regular—see Ex. I.51). 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: on the board)

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

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

(example on the board)

Seen:

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

Seen:

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

Open:

- Relation between context-free and regular languages

- 1 Context-Free Grammars and Languages
- 2 Context-Free and Regular Languages
- 3 The Word Problem for Context-Free Languages
- 4 The Emptiness Problem for CFLs
- 5 Pushdown Automata
- 6 Closure Properties of CFLs
- 7 Outlook

Theorem II.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 the class of CFLs.)

Theorem II.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 the class 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 recognizes L . $G := \langle N, \Sigma, P, S \rangle$ is defined as follows:

- $N := Q$, $S := q_0$
- if $\delta(q, a) = q'$, then $q \rightarrow aq' \in P$
- if $q \in F$, then $q \rightarrow \varepsilon \in P$

Obviously a w -labeled 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 A counterexample is $\{a^n b^n \mid n \geq 1\}$ (see Ex. I.51 and II.6).

Seen:

- CFLs are more expressive than regular languages

Seen:

- CFLs are more expressive than regular languages

Open:

- Decidability of word problem

- 1 Context-Free Grammars and Languages
- 2 Context-Free and Regular Languages
- 3 The Word Problem for Context-Free Languages
- 4 The Emptiness Problem for CFLs
- 5 Pushdown Automata
- 6 Closure Properties of CFLs
- 7 Outlook

- **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
⇒ only **finitely many combinations** to be inspected

Definition II.8

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

$$A \rightarrow BC \quad \text{or} \quad A \rightarrow a.$$

Definition II.8

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

$$A \rightarrow BC \quad \text{or} \quad A \rightarrow a.$$

Example II.9

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

$S \rightarrow AB \mid AC$	(generates L)
$A \rightarrow a$	(generates $\{a\}$)
$B \rightarrow b$	(generates $\{b\}$)
$C \rightarrow SB$	(generates $\{a^n b^{n+1} \mid n \geq 1\}$)

Theorem II.10

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

Theorem II.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 $n > 2$



Proof of Theorem II.10 (continued).

Step 1: (only $A \rightarrow a$)

- ① let $N' := \{B_a \mid a \in \Sigma\}$
- ② let $P' := \{A \rightarrow \alpha' \mid A \rightarrow \alpha \in P\} \cup \{B_a \rightarrow a \mid a \in \Sigma\}$
where α' is obtained from α by replacing every $a \in \Sigma$ with B_a

This yields G' (example: on the board)

Proof of Theorem II.10 (continued).

Step 1: (only $A \rightarrow a$)

- ① let $N' := \{B_a \mid a \in \Sigma\}$
- ② let $P' := \{A \rightarrow \alpha' \mid A \rightarrow \alpha \in P\} \cup \{B_a \rightarrow a \mid a \in \Sigma\}$
where α' is obtained from α by replacing every $a \in \Sigma$ with B_a

This yields G' (example: on the board)

Step 2: (elimination of $A \rightarrow B$)

- ① determine all derivations $A_1 \Rightarrow \dots \Rightarrow A_n$ with rules of the form $A \rightarrow B$ without repetition of nonterminals (\Rightarrow only finitely many!)
- ② let $P'' := (P \cup \{A_1 \rightarrow \alpha \mid A_1 \Rightarrow \dots \Rightarrow A_n \Rightarrow \alpha, \alpha \notin N\}) \setminus \{A \rightarrow B \mid A \rightarrow B \in P'\}$

This yields G'' (example: on the board)

Proof of Theorem II.10 (continued).

Step 3: for every $A \rightarrow A_1 \dots A_n$ with $n > 2$:

- ① add new symbols B_1, \dots, B_{n-2} to N''
- ② replace $A \rightarrow A_1 \dots A_n$ by

$$\begin{array}{rcl} A & \rightarrow & A_1 B_1 \\ B_1 & \rightarrow & A_2 B_2 \\ & \vdots & \\ B_{n-3} & \rightarrow & A_{n-2} B_{n-2} \\ B_{n-2} & \rightarrow & A_{n-1} A_n \end{array}$$

This yields G''' (example: on the board)

One can show: G, G', G'', G''' are equivalent

□

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 $N_{i,j} := \{A \in N \mid A \Rightarrow^* w[i, j]\}$
- ⑥ test whether $S \in N_{1,n}$ (and thus, whether $S \Rightarrow^* w[1, n] = w$)

The CYK Algorithm I

Algorithm II.11 (CYK Algorithm)

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

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

Procedure: **for** $i := 1$ **to** n **do**

$N_{i,i} := \{A \in N \mid A \rightarrow a_i \in P\}$

next i

for $d := 1$ **to** $n - 1$ **do** % compute $N_{i,i+d}$

for $i := 1$ **to** $n - d$ **do**

$j := i + d; N_{i,j} := \emptyset;$

for $k := i$ **to** $j - 1$ **do**

$N_{i,j} := N_{i,j} \cup \{A \in N \mid \text{there is } A \rightarrow BC \in P$

with $B \in N_{i,k}, C \in N_{k+1,j}\}$

next k

next i

next d

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

Example II.12

- $G : S \rightarrow SA \mid a$
 $A \rightarrow BS$
 $B \rightarrow BB \mid BS \mid b \mid c$
- $w = abaaba$
- Matrix representation of $N_{i,j}$

(on the board)

Seen:

- Word problem decidable using CYK algorithm

Seen:

- Word problem decidable using CYK algorithm

Open:

- Emptiness problem

- 1 Context-Free Grammars and Languages
- 2 Context-Free and Regular Languages
- 3 The Word Problem for Context-Free Languages
- 4 The Emptiness Problem for CFLs
- 5 Pushdown Automata
- 6 Closure Properties of CFLs
- 7 Outlook

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 Productivity Test

Algorithm II.13 (Productivity 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 $A \rightarrow \alpha \in P$ 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 Productivity Test

Algorithm II.13 (Productivity 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 $A \rightarrow \alpha \in P$ 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 II.14

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

(on the board)

Seen:

- Emptiness problem decidable using productivity test

Seen:

- Emptiness problem decidable using productivity test

Open:

- Characterizing automata model

- 1 Context-Free Grammars and Languages
- 2 Context-Free and Regular Languages
- 3 The Word Problem for Context-Free Languages
- 4 The Emptiness Problem for CFLs
- 5 Pushdown Automata
- 6 Closure Properties of CFLs
- 7 Outlook

- **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 specific bottom symbol
 - adding push and pop operations to transitions

Definition II.15

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**

Definition II.15

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' .

Definition II.16

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 $F \times \Gamma^* \times \{\varepsilon\}$.
- 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 $(q_0, Z_0, w) \vdash^* (q, \gamma, \varepsilon)$ for some $q \in F$, $\gamma \in \Gamma^*$.
- 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-recognizable** if $L = L(\mathfrak{A})$ for some PDA \mathfrak{A} .
- Two PDA $\mathfrak{A}_1, \mathfrak{A}_2$ are called **equivalent** if $L(\mathfrak{A}_1) = L(\mathfrak{A}_2)$.

Example II.17

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

Example II.17

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

Example II.17

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

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

Definition II.18

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

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

Definition II.18

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

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

Corollary II.19

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

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

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

Example II.20

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

Theorem II.21

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

Theorem II.21

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

Proof.

\Leftarrow omitted

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

- \mathfrak{A}_G simulates a derivation of G where the leftmost nonterminal of a sentence form 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



Proof of Theorem II.21 (continued).

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

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



Proof of Theorem II.21 (continued).

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

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

□

Example II.22

“Bracket language”, given by G :

$$S \rightarrow \langle \rangle \mid \langle S \rangle \mid SS$$

(on the board)

Seen:

- Definition of PDA
- Equivalence of PDA-recognizable and context-free languages

Seen:

- Definition of PDA
- Equivalence of PDA-recognizable and context-free languages

Open:

- Closure and decidability properties of CFLs

- 1 Context-Free Grammars and Languages
- 2 Context-Free and Regular Languages
- 3 The Word Problem for Context-Free Languages
- 4 The Emptiness Problem for CFLs
- 5 Pushdown Automata
- 6 Closure Properties of CFLs
- 7 Outlook

Theorem II.23

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

Theorem II.23

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

Theorem II.23

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
 $P := \{S \rightarrow S_1 S_2\} \cup P_1 \cup P_2$ generates $L_1 \cdot L_2$;

Theorem II.23

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

Theorem II.23

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



Theorem II.24

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

Theorem II.24

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).

Theorem II.24

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 would be 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			
	$w \in L$	$L = \emptyset$	$L_1 = L_2$
Reg	+ (I.38)	+ (I.40)	+ (I.42)
CFL	+ (II.11)	+ (II.13)	-

Overview of Decidability and Closure Results

Decidability Results			
	$w \in L$	$L = \emptyset$	$L_1 = L_2$
Reg	+ (I.38)	+ (I.40)	+ (I.42)
CFL	+ (II.11)	+ (II.13)	-

Closure Results					
	$L_1 \cdot L_2$	$L_1 \cup L_2$	$L_1 \cap L_2$	\overline{L}	L^*
Reg	+ (I.28)	+ (I.18)	+ (I.16)	+ (I.14)	+ (I.29)
CFL	+ (II.23)	+ (II.23)	- (II.24)	- (II.24)	+ (II.23)

- 1 Context-Free Grammars and Languages
- 2 Context-Free and Regular Languages
- 3 The Word Problem for Context-Free Languages
- 4 The Emptiness Problem for CFLs
- 5 Pushdown Automata
- 6 Closure Properties of CFLs
- 7 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)