

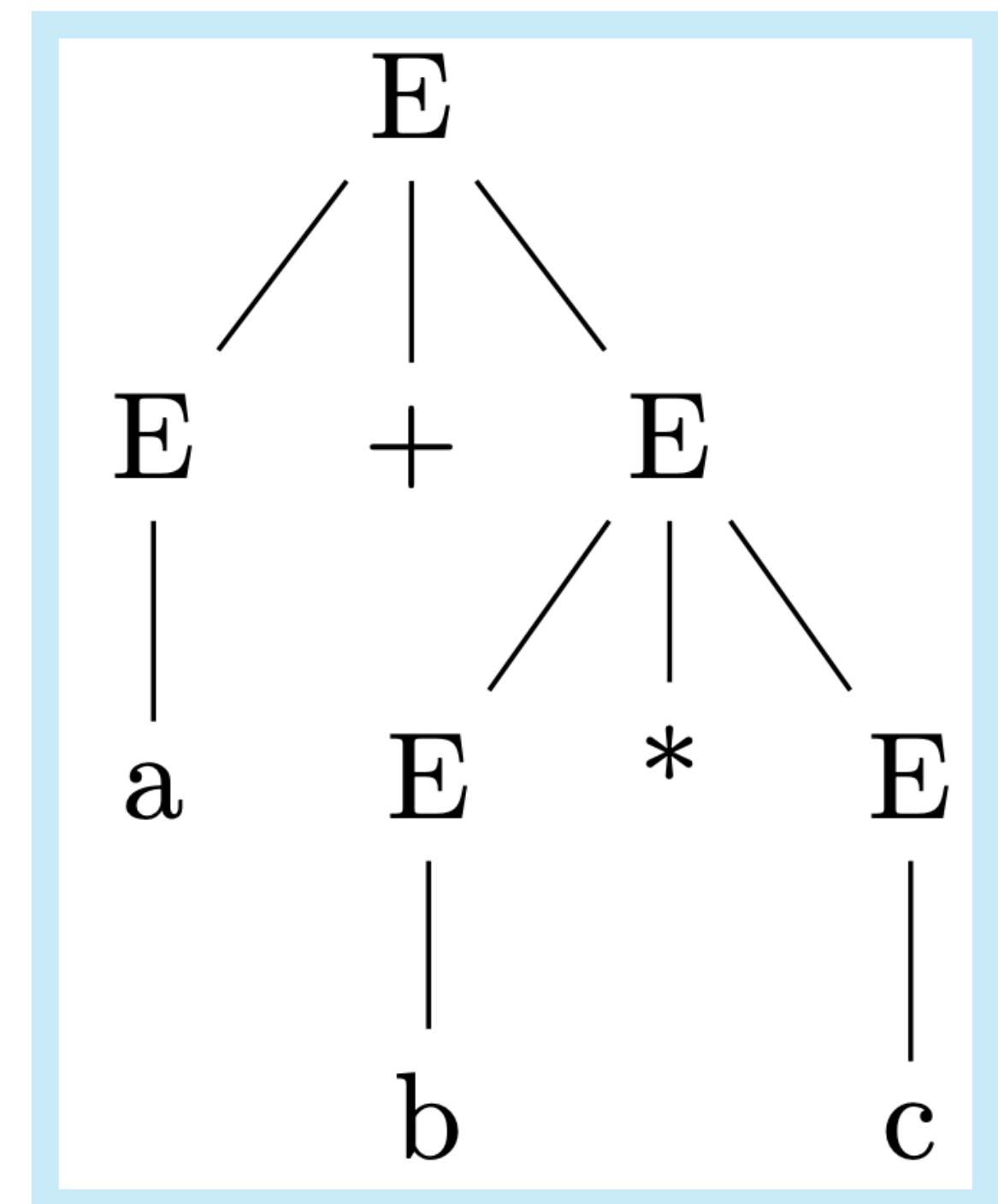
TDDD14 / TDDD85 – Lecture 8

CFG Rewriting and Normal Forms

August Ernstsson, 2026 (based on lecture notes by Jonas Wallgren)

Reminders: From last lecture

- CFG: context-free grammar
 - $G = \langle N, \Sigma, P, S \rangle$ (nonterminals, alphabet (terminals), productions, start symbol)
 - **Example:** $G = \langle \{E\}, \{a, b, c\}, P, E \rangle$
 - Where P is described by $E \rightarrow E^*E \mid E+E \mid a \mid b \mid c$ (set of productions)
- CFL: context-free language
 - $L(G)$ = language generated by grammar G
 - Superset of regular languages
- Derivations
 - **Example:** $E \Rightarrow E+E \Rightarrow E+E^*E \Rightarrow E+E^*c \Rightarrow E+b^*c \Rightarrow a+b^*c$
 - Derivation (parse) trees
 - Leftmost and rightmost derivations
- Ambiguous grammars



Leftmost or rightmost?

Reminders: Notational practice

- Nonterminals: *capital latin letters*
 - A, B, C, P, Q, S, T, ...
- Terminals: *small latin letters*
 - a, b, c, p, q, r, ...
- Strings of terminals and/or nonterminals: *small greek letters*
 - α , β , γ , ...

Let's start!

Introduction

- Rewriting grammars – Why?
 - **Simplification**
 - Minimize the number of productions
 - Efficiency
 - **Analysis:** Proving properties
 - **Implementation**, for example a parser
 - Parser generators
 - Some types of analysis or parser methods implementations may require a *grammar of a certain form!*

Simplification – Unnecessary symbols

- Consider a grammar over $\Sigma = \{a\}$
 - **Grammar G1**
 - $S \rightarrow AB \mid a$
 - $A \rightarrow a$
- **First step:** find out which nonterminals can produce strings.
 - Look "from right to left"
 - A: a string in Σ^* can be derived.
 - S: a string in Σ^* can be derived.
 - B: no string in Σ^* can be derived!

Simplification – Unnecessary symbols, cont.

- Consider a grammar over $\Sigma = \{a\}$
 - **Grammar G2**
 - $S \rightarrow a$
 - $A \rightarrow a$
- **Second step:** which nonterminals can be reached from start symbol S
 - Look "from left to right"
 - S can be reached from S (trivial)
 - A can't be reached from S!

Simplification – Unnecessary symbols, cont.

- Consider a grammar over $\Sigma = \{a\}$
 - **Grammar G_3**
 - $S \rightarrow a$
- All three grammars define the same language.
 - $L(G_1) = L(G_2) = L(G_3)$

Simplification – ϵ productions and unit productions

- *ϵ productions:* $A \rightarrow \epsilon$
- *Unit productions:* $A \rightarrow B$
- These rules can be convenient when defining a grammar.
- But: needlessly complicates analysis or implementation.

Simplification – ε productions and unit productions

- Example grammar over $\Sigma = \{ a, b, q, p \}$
 - **Grammar G_4**
 - $S \rightarrow aSb \mid T$
 - $T \rightarrow pTq \mid \varepsilon$
 - $L(G_4) = \{ a^m p^n q^n b^m \mid m \geq 0 \wedge n \geq 0 \}$.
- **If $A \rightarrow \alpha B \gamma$ and $B \rightarrow \varepsilon$ are in P' then put $A \rightarrow \alpha \gamma$ in P'**
- **If $A \rightarrow B$ and $B \rightarrow \beta$ are in P' then put $A \rightarrow \beta$ in P'**

Simplification – ε productions and unit productions

- Start: $P' = P = \{ S \rightarrow aSb, S \rightarrow T, T \rightarrow pTq, T \rightarrow \varepsilon \}$ (4 productions)
- Extension steps:
 - Epsilon: **If** $A \rightarrow \alpha B \gamma$ **and** $B \rightarrow \varepsilon$ **are in** P' **then** put $A \rightarrow \alpha \gamma$ **in** P'
 - Unit: **If** $A \rightarrow B$ **and** $B \rightarrow \beta$ **are in** P' **then** put $A \rightarrow \beta$ **in** P'
- Application:
 - **Since** $S \rightarrow T$ **and** $T \rightarrow \varepsilon$ **in** P' , **then** put $S \rightarrow \varepsilon$ **in** P' (unit or epsilon)
 - **Since** $T \rightarrow pTq$ **and** $T \rightarrow \varepsilon$ **in** P' , **then** put $T \rightarrow pq$ **in** P' (epsilon)
 - **Since** $S \rightarrow aSb$ **and** $S \rightarrow \varepsilon$ **in** P' , **then** put $S \rightarrow ab$ **in** P' (epsilon)
 - **Since** $S \rightarrow T$ **and** $T \rightarrow pTq$ **in** P' , **then** put $S \rightarrow pTq$ **in** P' (unit)
 - **Since** $S \rightarrow pTq$ **and** $T \rightarrow \varepsilon$ **in** P' , **then** put $S \rightarrow pq$ **in** P' (epsilon)

Simplification – ε productions and unit productions

- Finally, remove all ε productions and unit productions from P' (3 of them).
- **Grammar $G_5 = \langle N, \Sigma, P', S \rangle$**
 - $S \rightarrow aSb \mid ab \mid pTq \mid pq$
 - $T \rightarrow pTq \mid pq$
 - $S \rightarrow \varepsilon$
- **But:** $\varepsilon \in L(G_4)$
 - $L(G_5) = L(G_4) - \{\varepsilon\}$
 - We need to add $S \rightarrow \varepsilon$ to G_5 .
 - Now $L(G_5) = L(G_4)$.

- **Grammar G_4**
 - $S \rightarrow aSb \mid T$
 - $T \rightarrow pTq \mid \varepsilon$

Definition 1: Chomsky normal form

- A grammar is in Chomsky normal form
 - if all rules have the form $A \rightarrow a$ or $A \rightarrow BC$.
- Example: **Grammar G6**
 - $S \rightarrow aSb \mid ab \mid pTq \mid pq$
 - $T \rightarrow pTq \mid pq$

Chomsky normal form: Step 1

- Insert productions for every terminal ($\Sigma = \{ a, b, q, p \}$)
 - $A \rightarrow a, \quad B \rightarrow b, \quad Q \rightarrow q, \quad P \rightarrow p$
- Then update existing productions to replace the terminals
- **Grammar G_7**
 - $S \rightarrow ASB \mid AB \mid PTQ \mid PQ$
 - $T \rightarrow PTQ \mid PQ$

Chomsky normal form: Step 2

- Then for all productions of the form $A \rightarrow B_1B_2B_3\dots$
 - replace it by $A \rightarrow B_1C$ and $C \rightarrow B_2B_3\dots$
 - Iterate as many steps as necessary.

- **Grammar G7** (same as last slide)

- $S \rightarrow ASB \mid AB \mid PTQ \mid PQ$
- $T \rightarrow PTQ \mid PQ$
- $L(G8) = L(G7) = L(G6) = L(G4) - \{\epsilon\}$

- **Grammar G8**

- $S \rightarrow AE \mid AB \mid PF \mid PQ$
- $T \rightarrow PF \mid PQ$
- $E \rightarrow SB$
- $F \rightarrow TQ$
- $A \rightarrow a$
- $B \rightarrow b$
- $P \rightarrow p$
- $Q \rightarrow q$

Definition 1: Greibach normal form

- A grammar is in Greibach normal form (GNF)
 - **if all** productions have the form $A \rightarrow aB_1B_2B_3\dots$
 - (Sequence of B's may be empty)
- Every CFG can be converted to a CFG in GNF.
- Greibach normal form will be important in lecture 10.

Left recursion

- Left recursive property:
 - In *at least one* production for *at least one* nonterminal,
 - the *same* nonterminal occurs *first* in the right-hand side.
- **Example**: the productions
 - $A \rightarrow Ap \mid q$
- Possible derivation
 - $A \Rightarrow Ap \Rightarrow App \Rightarrow Appp \Rightarrow qppp$

Left recursion, cont.

- Consider: a (recursive descent) parser strictly following the grammar rules
 - Invokes the parsing procedure for A (Aproc)
 - $A \rightarrow Ap$: invoke Aproc again
 - $A \rightarrow Ap$: invoke Aproc again
 - $A \rightarrow Ap$: invoke Aproc again
 - ...
 - There is infinite recursion!
 - The parser is never given the chance to look for p or q.

Left recursion – Solution

- **Solution:** rewrite grammar!
- We want derivations on the form $B \Rightarrow qC \Rightarrow \dots$
 - Always "consume" a terminal from the input string.
 - Input is finite, so this guarantees termination.
- Substitute left-recursive nonterminals A with A'
- For our example:
 - $A \rightarrow qA'$
 - $A' \rightarrow pA' \mid \varepsilon$

Left recursion – Another example

- Another left-recursive set of productions:
 - $A \rightarrow Ap \mid Aq \mid Ar \mid a \mid b \mid c$
 - There are multiple cases of left-recursion.
 - We have to handle them all at once.
- Rewritten set of productions:
 - $A \rightarrow aA' \mid bA' \mid cA'$
 - $A' \rightarrow pA' \mid qA' \mid rA' \mid \varepsilon$

Left recursion

- *Mutual* recursion:
 - a rule for A begins with B and a rule for B begins with A
- There are also more complicated cases.
 - All grammars can be rewritten to non-leftrecursive form.

Coming up

- This week
 - **Monday:** Context-free grammars (CFG) introduction
 - **Wednesday:** CFG rewriting, GFG normal forms **Done!**
- Next week
 - **Wednesday:** Pushdown automata (PDA)
 - **Friday:** Equivalence between CFG and PDA
- Later: properties of CFGs and parsing methods for CFGs

Thanks for today!