TDDD14 / TDDD85 – Lecture 8

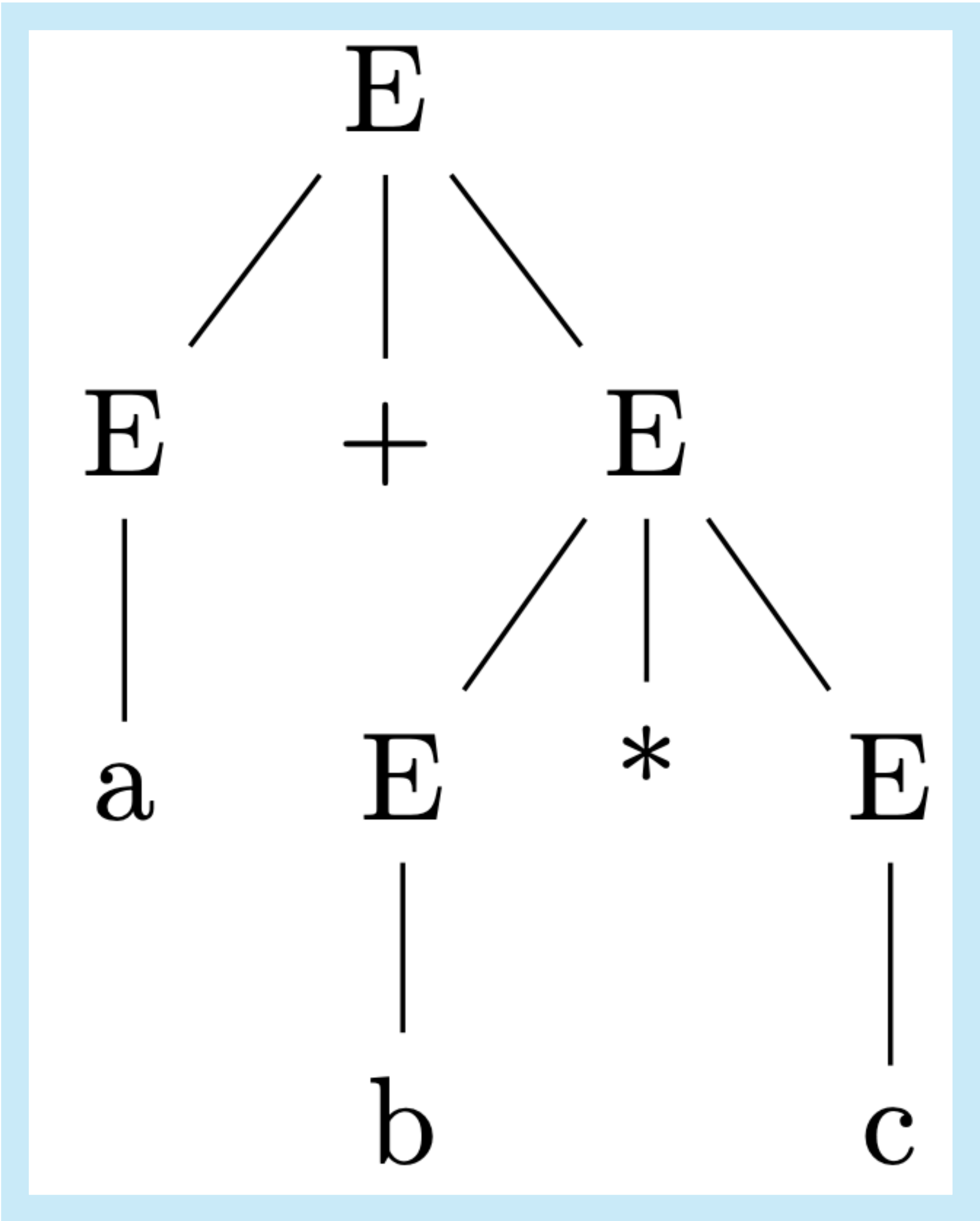
CFG Rewriting and Normal Forms

August Ernstsson, 2024 (based on lecture notes by Jonas Wallgren)

LINKÖPING UNIVERSITY

Reminders: From last lecture

- CFG: context-free grammar
  - G = <N,Σ,P,S> (nonterminals, alphabet (terminals), productions, start symbol)
  - **Example**: G = <{E}, {a,b,c}, P, E>
    - 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

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

Ambiguous Grammar Example

- On the whiteboard: example of two different leftmost derivations for the same string

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 grammar of a certain form

Simplification – Unnecessary symbols

- Consider a grammar over $\Sigma = \{a\}$
  - **Grammar G1**
  - $S \rightarrow AB|a$
  - $A \rightarrow a$

- **First step**: find out which nonterminals can produce strings.
  - Look ”from right to left”
  - A: a string in $\Sigma*$ can be derived.
  - S: a string in $\Sigma*$ can be derived.
  - B: no string in $\Sigma*$ 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 G3**
  - $S \rightarrow a$

- All three grammars define the same language.
  - L(G1) = L(G2) = L(G3)

Simplification – ε productions and unit productions

- *ε productions*: $A \rightarrow \varepsilon$
- *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 G4**
    - $S \rightarrow aSb \mid T$
    - $T \rightarrow pTq \mid \varepsilon$
  - $L(G4) = \{ a^m p^n q^n b^m \mid m \geq 0 \wedge n \geq 0 \}$.

- **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:
  - **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'
- Application: (colored by which extension step is used)
  - **Since** $S \rightarrow T$ **and** $T \rightarrow \varepsilon$ **in** P', **put** $S \rightarrow \varepsilon$ **in** P'
  - **Since** $T \rightarrow pTq$ **and** $T \rightarrow \varepsilon$ **in** P', **put** $T \rightarrow pq$ **in** P'
  - **Since** $S \rightarrow aSb$ **and** $S \rightarrow \varepsilon$ **in** P', **put** $S \rightarrow ab$ **in** P'
  - **Since** $S \rightarrow T$ **and** $T \rightarrow pTq$ **in** P', **put** $S \rightarrow pTq$ **in** P'
  - **Since** $S \rightarrow pTq$ **and** $T \rightarrow \varepsilon$ **in** P', **put** $S \rightarrow pq$ **in** P'

(Note: First row matches both types of extension. This was what caused some confusion during the lecture.)

Simplification – ε productions and unit productions

- Finally, remove all ε productions and unit productions from P' (3 of them).

- **Grammar G5** = < N, Σ, P', S>
  - $S \rightarrow aSb \mid ab \mid pTq \mid pq$
  - $T \rightarrow pTq \mid pq$
  - $S \rightarrow \varepsilon$

- **But**: $\varepsilon \in L(G4)$
  - $L(G5) = L(G4) - \{\varepsilon\}$
  - We need to add $S \rightarrow \varepsilon$ to G5.
    - Now $L(G5) = L(G4)$.

- **Grammar G4**
  - $S \rightarrow aSb \mid T$
  - $T \rightarrow pTq \mid \varepsilon$

Definition 1: Chomsky normal form

- A grammar is in <u>Chomsky normal form</u>
  - 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 ($\Sigma = \{ a, b, q, p \}$)
  - $A \to a, \quad B \to b, \quad Q \to q, \quad P \to p$

- Then update existing productions to replace the terminals
  - **Grammar G7**
    - $S \to ASB \mid AB \mid PTQ \mid PQ$
    - $T \to PTQ \mid PQ$

Chomsky normal form: Step 2

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

- **Grammar G7** (same as last slide)
  - S → ASB | AB | PTQ | PQ
  - T → PTQ | PQ

- L(G8) = L(G7) = L(G6) = L(G4) - {ε}

- **Grammar G8**
  - S → AE |AB | PF | PQ
  - T → PF | PQ
  - E → SB
  - F → TQ
  - A → a
  - B → b
  - P → p
  - Q → q

Definition 1: Greibach normal form

- A grammar is in Greibach normal form (GNF)
  - **if all** productions have the form $A \rightarrow aB_1B_2B_3\ldots$
  - (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

- <u>Left recursive property</u>:
  - 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

- Consider: a (recursive descent) parser strictly following the grammar rules
  - Invokes the parsing procedure for A (Aproc)
  - A → Ap: invoke Aproc again
  - A → Ap: invoke Aproc again
  - A → Ap: invoke Aproc again
  - …
  - There is infinite recursion!
    - as 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 $B \Rightarrow qC \Rightarrow \cdots$
  - 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 → Ap | Aq | Ar | a | b | c
  - There are multiple cases of left-recursion.
    - We have to handle them all at once.

- Rewritten set of productions:
  - A → aA' | bA' | cA'
  - A' → pA' | qA' | rA' | ε

Left recursion

- *Mutual* recursion:
  - a rule for A begins with B and a rule for B begins with A
- There could also be yet more complicated cases.
- But all such recursivity is solvable.
- All grammars can be rewritten to non-leftrecursive form.

To think about

- How do the various rewriting methods affect the number of productions?
  - Could there e.g. be exponential blow-up?

- Give a grammar in Chomsky normal form for
  - the language $\{ a^n b^{2n} c^k \mid n \geq 1 \land k \geq 1 \}$

Coming up

- This week
  - **Today**: Context-free grammars (CFG) introduction
  - **Wednesday**: CFG rewriting, GFG normal forms

Done!

- Next week
  - **Monday**: Pushdown automata (PDA)
  - **Friday**: Equivalence between CFG and PDA

- Then three weeks with one lecture per week
  - Properties of CFGs and parsing methods for CFGs

Thanks for today!

LINKÖPING UNIVERSITY