CSE 555 Theory of Computation Class 8 (2/7)

Goran Konjevod

Department of Computer Science and Engineering
Arizona State University

ASU, Spring 2008

Context-free languages

Regular language definitions:

- accepted by DFAs
- represented by R.E.s
- generated by regular grammars

What is a regular grammar?

1. Finite alphabet: $\Sigma = \{a_1, \ldots, a_m\}$
2. Finite set of variables: $V = \{S, A_1, A_2, A_3, \ldots, A_n\}$
3. Finite set of rules: $A_i \rightarrow a_j A_k$ or $A_i \rightarrow a_j$.
4. Generative process:
    1. Starting from string "S",
    2. Repeatedly transform the current string by applying rules
    3. Stop when the string consists only of symbols from Σ (*terminals*)

A grammar *generates a language.*

Regular grammars and regular languages

Any regular grammar generates a regular language.
Any regular language can be generated by a regular grammar.

(Proof: a direct correspondence between reg. grammars and NFAs.)

More general grammars

1. Finite alphabet: $\Sigma = \{a_1, \ldots, a_m\}$
2. Finite set of variables: $V = \{S, A_1, A_2, A_3, \ldots, A_n\}$
3. Finite set of rules.
4. Generative process:
    1. Starting from string "S",
    2. Repeatedly transform the current string by applying rules
    3. Stop when the string consists only of symbols from Σ (*terminals*)

A grammar *generates a language.*

Context-free grammars (CFG)

1. Finite alphabet: $\Sigma = \{a_1, \ldots, a_m\}$
2. Finite set of variables: $V = \{S, A_1, A_2, A_3, \ldots, A_n\}$
3. Finite set of rules: $A_i \to u$, $u \in (\Sigma \cup V)^*$.
4. Generative process:
    1. Starting from string "S",
    2. Repeatedly transform the current string by applying rules
    3. Stop when the string consists only of symbols from Σ (*terminals*)

A grammar *generates a language.*

Context-sensitive grammars (CSG)

1. Finite alphabet: $\Sigma = \{a_1, \ldots, a_m\}$
2. Finite set of variables: $V = \{S, A_1, A_2, A_3, \ldots, A_n\}$
3. Finite set of rules: $uAv \longrightarrow uxv$, $u, v \in (\Sigma \cup V)^*$ and $x \in (\Sigma \cup V)^+$, or the rule $S \longrightarrow \epsilon$.
4. Generative process:
    1. Starting from string "S",
    2. Repeatedly transform the current string by applying rules
    3. Stop when the string consists only of symbols from Σ (*terminals*)

A grammar *generates a context-sensitive language.*
(Alternative definition: for every non-zero rule, the right-hand side is at least as long as the left-hand side.)

General grammars

1. Finite alphabet: $\Sigma = \{a_1, \dots, a_m\}$
2. Finite set of variables: $V = \{S, A_1, A_2, A_3, \dots, A_n\}$
3. Finite set of rules: $uAv \rightarrow x$, $u, v, x \in (\Sigma \cup V)^*$.
4. Generative process:
    1. Starting from string "S",
    2. Repeatedly transform the current string by applying rules
    3. Stop when the string consists only of symbols from Σ (*terminals*)

The class of languages: recursively enumerable languages.

Examples

CFG for $L_1 = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$?

CFG for $L_2 = \{w \in \{a, b\}^* \mid n_a(w) \leq n_b(w)\}$?

CFG for $L_3 = \{w \in \{a, b\}^* \mid n_a(w) = 2n_b(w)\}$?

CFG for $L_4 = \{w \in \{a, b\}^* \mid n_b(w) \leq n_a(w) \leq 2n_b(w)\}$?

CFG for $L_5 = \{w \in \{a, b\}^* \mid n_b(w) < n_a(w) \leq 2n_b(w)\}$?

The parsing problem

Given a grammar G and a string u, is u derivable from G?

Idea: try to explore all possible derivations.
Problem: we don't know when to stop and report "No."

Idea: simplify the grammar so that this becomes easy.
Solution: reduce G to Chomsky normal form.

Chomsky normal form (CNF)

A CFG G is in Chomsky normal form, if all its rules have one of these forms:

$$A \longrightarrow BC$$
$$A \longrightarrow a$$
$$S \longrightarrow \epsilon$$

Every CFG has an equivalent CNF grammar.

Chomsky normal form reduction

1. Add new start symbol S_0 and the rule $S_0 \longrightarrow S$. (S_0 will never occur on the right-hand side.)
2. Remove $A \longrightarrow \epsilon$ for $A \neq S$. For each rule $R \longrightarrow uAv$, add $R \longrightarrow uv$. (E.g. $R \longrightarrow uAvAw$ results in $R \longrightarrow uAvw$ and $R \longrightarrow uvAw$ and $R \longrightarrow uvw$.)
   If there was a rule $R \longrightarrow A$, add $R \longrightarrow \epsilon$ unless we've removed such a rule already.
3. Repeat previous step until no more ϵ-rules (except $S \longrightarrow \epsilon$.)
4. Replace $A \longrightarrow u_1 u_2 \cdots u_k$, $k \geq 3$ with the rules $A \longrightarrow u_1 A_1$, $A_1 \longrightarrow u_2 A_2$, ..., $A_{k-2} \longrightarrow u_{k-1} u_k$. In any new rule $A_i \longrightarrow u_{i+1} A_{i+1}$, replace u_{i+1} by variable U_{i+1} and add rule $U_{i+1} \longrightarrow u_{i+1}$.