Automata and Grammars (BIE-AAG)

11. Context-free Grammars

Jan Holub

Department of Theoretical Computer Science

Faculty of Information Technology

Czech Technical University in Prague

Context-free Grammars

- Context-free grammars
  - They can describe most of the syntactic structures of programming languages.
  - Algorithms for effective analysis of sentences of context-free languages are known.
- Syntactic analysis:
  - Is the given string w generated by grammar G?
  - What is the structure of the string?

Ambiguous and Unambiguous Grammars

Definition

Context-free grammar $G = (N, T, P, S)$ is *ambiguous*, if there is a sentence $w \in L(G)$ that is a result of at least two different derivation trees. Otherwise the grammar is *unambiguous*.

Ambiguous and Unambiguous Grammars

Example

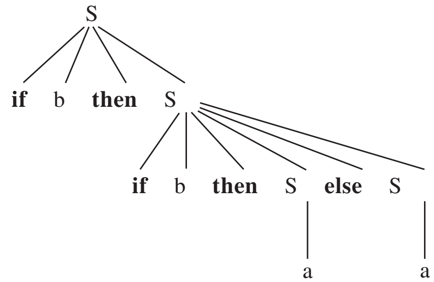
$G = (\{S\},\{a,b,\mathbf{if},\mathbf{then},\mathbf{else}\},P,S)$, kde P:

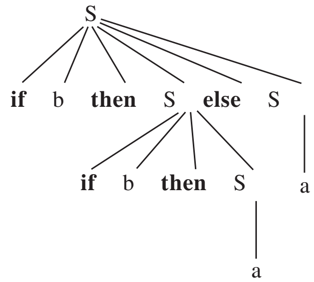
$S \rightarrow \mathbf{if}\ b\ \mathbf{then}\ S\ \ \mathbf{else}\ S$

$S \rightarrow \mathbf{if}\ b\ \mathbf{then}\ S$

$S \rightarrow\ a$

The grammar is ambiguous: $\mathbf{if}\ b\ \mathbf{then}\ \mathbf{if}\ b\ \mathbf{then}\ a\ \mathbf{else}\ a$





Ambiguous and Unambiguous Grammars

Example
CFG contains a rule $A \to AA$, therefore it is ambiguous. For a sentential form AAA there are two different derivation trees.

Ambiguousness can be removed by replacing $A \to AA$ with rules
$A \to AB$
$A \to B$. □

Ambiguous and Unambiguous Grammars

- Ambiguousness concerns the grammar only
- Inherently ambiguous languages:
  Cannot be generated by an unambiguous grammar.
- It is impossible to create an algorithm that would decide whether the given context-free grammar is ambiguous or that could modify such grammar to get an unambiguous grammar (by reduction from Post's correspondence problem).

Example
$G = (\{S_1, S_2\}, \{a, b, \mathbf{if}, \mathbf{then}, \mathbf{else}\}, P, S_1)$, kde P:
$S_1 \to \mathbf{if}\ b\ \mathbf{then}\ S_1 \mid \mathbf{if}\ b\ \mathbf{then}\ S_2\ \mathbf{else}\ S_1 \mid a$
$S_2 \to \mathbf{if}\ b\ \mathbf{then}\ S_2\ \mathbf{else}\ S_2 \mid a$
Symbol $\mathbf{else}$ always belongs to the closest $\mathbf{then}$.

Transformations of CF grammars

Theorem
There is an algorithm that decides whether a language generated by the given context-free grammar is empty. □

Algorithm Is $L(G)$ non-empty?
Input: Context-free grammar $G = (N, T, P, S)$.
Output: Yes if $L(G) \neq \emptyset$, no otherwise.
Method: We create sets $N_0, N_1, \ldots$ thusly:

1. $N_0 = \emptyset$, $i := 1$
2. $N_i = \{A : A \to \alpha \in P, \alpha \in (N_{i-1} \cup T)^*\} \cup N_{i-1}$
3. If $N_i \neq N_{i-1}$, then $i := i + 1$ and perform (2), otherwise $N_t = N_i$.
4. If $S \in N_t$, then yes, otherwise no. □

Transformations of CF grammars

Example
$G = (\{S, A, B\}, \{a, b\}, \{S \to a, S \to A, A \to AB, B \to b\}, S)$.

$N_0 = \emptyset$
$N_1 = \{S, B\}$
$N_2 = \{S, B\}$
$N_1 = N_2 = N_t$
$S \in N_t \Rightarrow$ grammar G generates a non-empty language.

Transformations of CF grammars

Definition
Symbol $X \in N \cup T$ is *unreachable* in context-free grammar $G = (N, T, P, S)$, if X does not appear in any sentential form, that is there is no derivation of the form $S \Rightarrow^{+} \alpha X \beta,\ \alpha, \beta \in (N \cup T)^{*}$.

Transformations of CF grammars

Algorithm Exclusion of unreachable symbols
Input: Context-free grammar $G = (N, T, P, S)$.
Output: $G' = (N', T', P', S)$ such that
a) $L(G') = L(G)$,
b) for all $X \in N' \cup T'$ there exist $\alpha, \beta \in (N' \cup T')^{*}$ such that $S \Rightarrow^{*} \alpha X \beta$.
Method:

1. $V_0 = \{S\}$ and $i := 1$.
2. $V_i = \{X : A \rightarrow \alpha X \beta \in P, A \ \in V_{i-1}\} \cup V_{i-1}$.
3. If $V_i \neq V_{i-1}$, then $i := i + 1$ and perform step 2, otherwise

$N' = V_i \cap N$,
$T' = V_i \cap T$,
$P' = \{A \rightarrow \alpha : A \ \in N', \alpha \in V_i{}^{*}, A \ \rightarrow \alpha \in P\}$,
$G' = (N', T', P', S)$

Transformations of CF grammars

Example
$G = (\{S, A, B\}, \{a, b\}, \{S \rightarrow a, S \ \rightarrow A, A \ \rightarrow AB, B \rightarrow b\}, S)$

$V_0 = \{S\}$
$V_1 = \{S\} \cup \{a, A\}$
$V_2 = \{S, A, a\} \cup \{B\}$
$V_3 = \{S, A, B, a, \} \cup \{b\}$
$V_4 = \{S, A, B, a, b\}$

Transformations of CF grammars

Definition
Symbol $X \in N \cup T$ is *redundant* in $G = (N, T, P, S)$, if there does not exist
$S \Rightarrow^{*} wXy \Rightarrow^{*} wxy$, where $w, x, y \in T^{*}$.

Transformations of CF grammars

Algorithm Exclusion of redundant symbols.
Input: CFG $G = (N, T, P, S)$, $L(G) \neq \emptyset$.
Output: CFG $G' = (N', T', P', S)$,
a) $L(G') = L(G)$ and
b) $\forall X \in N' \cup T'\ S \Rightarrow^* \alpha X \beta$.
Method:

1. Using algorithm 3.6 (is $L(G)$ empty?) we get N_t.
   $G_1 = (N_t, T, P_1, S)$,
   $P_1 = \{A \rightarrow \alpha : A \in N_t, \alpha \in (N_t \cup T)^*, A \rightarrow \alpha \in P\}$.
2. Algorithm 3.9 (Exclusion of unreachable symbols) excludes all nonterminals from G that cannot generate terminal string and then all unreachable symbols. We get $G' = (N', T', P', S)$.

Transformations of CF grammars

Example
$G = (\{S, A, B\}, \{a, b\}, \{S \rightarrow a, S \rightarrow A, A \rightarrow AB, B \rightarrow b\}, S)$

Step 1.:
$N_t = \{S, B\}$
$G_1 = (\{S, B\}, \{a, b\}, \{S \rightarrow a, B \rightarrow b\}, S)$
Step 2.:
$V_0 = \{S\}$
$V_1 = \{S, a\}$
$V_2 = \{S, a\}$
$G' = (\{S\}, \{a\}, \{S \rightarrow a\}, S)$

Transformations of CF grammars

Theorem
(Rule exclusion theorem, substitution theorem).
$G = (N, T, P, S)$ is a context-free grammar and
$A \rightarrow \alpha B \beta \in P, B \in N, A \neq B$ and $\alpha, \beta \in (N \cup T)^*$.
Let $B \rightarrow \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_k$ be all rules in P with symbol B on the left-hand side. Let $G' = (N, T, P', S)$, where
$P' = P \cup \{A \rightarrow \alpha\gamma_1\beta \mid \alpha\gamma_2\beta \mid \ldots \mid \alpha\gamma_k\beta\} \setminus \{A \rightarrow \alpha B \beta\}$. Then $L(G) = L(G')$.

Transformations of CF grammars

Definition
Context-free grammar $G = (N, T, P, S)$ is *cycle-free*, if the following derivation: $A \Rightarrow^+ A$ is not possible for any $A \in N$. $\square$

Definition
Context-free grammar $G = (N, T, P, S)$ is *proper*, if it has no cycles, no ε-rules, no unreachable sybmols and no redundant symbols. $\square$

Theorem
If context-free grammar $G = (N, T, P, S)$ has no ε-rules and simple rules, then it is cycle-free. $\square$

Theorem
If L is a context-free language, then it can be generated by some proper grammar G. $\square$

Transformations of CF grammars

Definition
Context-free grammar $G = (N, T, P, S)$ is *reduced*, if it does not contain redundant symbols. $\square$

Chomsky normal form

Definition
CFG $G = (N, T, P, S)$ is in Chomsky normal form if every rule in P is in one of the following forms:

1. $A \to BC$ for $A, B, C \in N$.
2. $A \to a$ for $a \in T, A \in N$.
3. If $\varepsilon \in L(G)$, then $S \to \varepsilon$ is a rule in P and S does not appear on the right-hand side of any of the rules. $\square$

Chomsky normal form

Algorithm Conversion of grammar to Chomsky normal form
Input: Proper context-free grammar $G = (N, T, P, S)$.
Output: CFG $G' = (N', T, P', S)$ in Chomsky normal form, $L(G) = L(G')$
Method:

1. $P' = \emptyset$.
2. Add all rules of form $A \to a$ from P into P'.
3. Add all rules of form $A \to BC$ from P into P'.
4. If $S \to \varepsilon$ is in P, add $S \to \varepsilon$ into P'.

Chomsky normal form

Algorithm (continued):

5. For every rule of form $A \to X_1 X_2 \dots X_k \in P$, where $k > 2$, add into P' the following rules (when $X_i \in N$, $X_i' = X_i$, when $X_i \in T$, X_i' is a new nonterminal):

$$\begin{aligned} A &\to X_1' \langle X_2 \dots X_k \rangle \\ \langle X_2 \dots X_k \rangle &\to X_2' \langle X_3 \dots X_k \rangle \\ &\vdots \\ \langle X_{k-2} \dots X_k \rangle &\to X_{k-1}' \langle X_{k-1} X_k \rangle \\ \langle X_{k-1} X_k \rangle &\to X_{k-1}' X_k', \end{aligned}$$

where every symbol $\langle X_i \dots X_k \rangle$ is a new nonterminal symbol in P.

Chomsky normal form

Algorithm (continued):

6. For every rule of the form $A \to X_1X_2$, where X_1 or X_2 or X_1 and X_2 are in T, add into P' rules $A \to X_1'X_2'$.
7. For every nonterminal symbol a' created in steps 5. and 6. add into P' rule $a' \to a$. Finally N' is the union of N with all new nonterminals. Grammar $G' = (N', T, P', S)$.

Chomsky normal form

Theorem
Let L be a context-free language. Then L is a language generated by some grammar in Chomsky normal form.

Chomsky normal form

Example
Proper CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$, P:
$S \to aAB \mid BA$
$A \to BBB \mid a$
$B \to AS \mid b$

step 1: $A \to a$
$B \to b$ is added into P'.
step 2: $S \to BA$
$B \to A\,S$ is added into P'.
step 3: We do nothing.
step 4: $S \to aAB \in P \Rightarrow S \to a'\langle AB\rangle$
$\langle AB\rangle \to AB$ is added into P'.
$A \to BBB \in P \Rightarrow A \to B\langle BB\rangle$
$\langle BB\rangle \to BB$ is added into P'.

Chomsky normal form

Example (continued)
step 5: We do nothing.
step 6: $a' \to a$ is added into P'.
P': $S \to a'\langle AB\rangle \mid BA$
$A \to B\langle BB\rangle \mid a$
$B \to AS \mid b$
$\langle AB\rangle \to AB$
$\langle BB\rangle \to BB$
$a' \to a$
$N' = N \cup \{\langle AB\rangle, \langle BB\rangle, a'\}$

Chomsky normal form

Algorithm Cocke-Younger-Kasami (CYK) algorithm
Input: Context-free grammar $G = (N, T, P, S)$ in Chomsky normal form, $X \in T^*$
Output: Answer, whether $X \in L(G)$
Method: $X = \{x_1, x_2, \ldots, x_n\}$

1. Initialize array $P[n, n]$ to $\emptyset$.
2. For every $i = 1..n$ and every rule $A \to t_i$ set $P[i, 1] := P[i, 1] \cup \{A\}$
3. For every $i = 2..n$ do (string length)
   For every $j = 1..n - i + 1$ do (position in text)
   For every $k = 1..i - 1$ do (length of generated string)
   If $B \in P[j, k]$, $C \in P[j + k, i - k]$ and $A \to BC$
   then $P[j, i] := P[j, i] \cup \{A\}$
4. If $S \in P[1, n]$, then $X \in L(G)$,
   otherwise $X \notin L(G)$.

Chomsky normal form

Example
Text $aaba$ and CFG $G = (\{S, A, B, C\}, \{a, b\}, P, S)$, P:
$S \to AB \mid BC$
$A \to BA \mid a$
$B \to CC \mid b$
$C \to AB \mid a$

4				
3				
2				
1				
	a	a	b	a

Chomsky normal form

Theorem
If $G = (N, T, P, S)$ is a CFG and $X \in T^*$ is a string of length n, then it is possible to find out whether $X \in L(G)$ in time $\mathcal{O}(n^3)$.

Greibach normal form

Definition
Nonterminal symbol A in CFG $G = (N, T, P, S)$ is called *recursive*, if there exists a derivation $A \Rightarrow^+ \alpha A \beta$ for some α and $\beta \in (N \cup T)^*$. If $\alpha = \varepsilon$, then A is called *left-recursive symbol*, similarly if $\beta = \varepsilon$, then A is called *right-recursive symbol*.
Grammar with at least one (right-)left-recursive nonterminal is called *(right-)left-recursive*.
Grammar in which at least one nonterminal symbol is recursive is called *recursive*.

Greibach normal form

Theorem
(Removing left recursion from the rules).
Let $G = (N, T, P, S)$ be a CFG, in which
$A \to A\,\alpha_1 \mid A\,\alpha_2 \mid \ldots \mid A\,\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$
are all rules in P with nonterminal symbol A on the left-hand side and no $\beta_i, i = 1, 2, \ldots, n$, begins with symbol A.
Let $G' = (N \cup \{A'\}, T, P', S)$ be a context-free grammar, where P' is the set P in which all above mentioned rules are replaced by these rules:
$A \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n \mid \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$
$A' \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m \mid \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A'$,
where A' is a new nonterminal symbol that is not in N. Then $L(G') = L(G)$.

Greibach normal form

Example
$G = (\{E, T, F\}, \{+, *, (,), a\}, P, E)$, P:
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid a$.

We remove left recursion:
$E \to T \mid TE'$
$E' \to +T \mid +TE'$
$T \to F \mid FT'$
$T' \to *F \mid *FT'$
$F \to (E) \mid a$
$G' = (\{E, E', T', T, F\}, \{+, *, (,), a\}, P', E)$, P' is the set of rules given above.

Greibach normal form

Definition
Context-free grammar G is in Greibach normal form if G has no ε-rules and every rule that does not contain an empty string on the right-hand side has form $A \to a\alpha$, where $a \in T, \alpha \in N^*$. $\square$

Greibach normal form

Algorithm Exclusion of left recursion.
Input: Proper context-free grammar $G = (N, T, P, S)$.
Output: CFG G' without left recursion, $L(G) = L(G')$.
Method:

1. We choose ordering $N = \{A_1, \ldots, A_r\}$: $A_i \to \alpha$, α begins either by a terminal symbol or nonterminal A_j for $j \geq i$.
   We set $i := 1$.

Greibach normal form

Algorithm (continued):

2. All rules $A_i \to A_i\alpha_1 \mid \dots \mid A_i\alpha_m \mid \beta_1 \mid \dots \mid \beta_n$ from P with nonterminal symbol A_i on the left-hand side, where no β_j begins with a nonterminal symbol A_k for $k \leq i$, are replaced by these rules:
   $A_i \to \beta_1 \mid \dots \mid \beta_n \mid \beta_1 A_i' \mid \dots \mid \beta_n A_i'$,
   $A_i' \to \alpha_1 \mid \dots \mid \alpha_m \mid \alpha_1 A_i' \mid \dots \mid \alpha_m A_i'$, where A_i' is a new nonterminal symbol.
3. If $i = r$, we have the desired grammar G' and exit, otherwise $i := i + 1$ and $j := 1$.
4. If for nonterminal symbol A_j there exist rules $A_j \to \beta_1 \mid \dots \mid \beta_m$, then we replace all rules of the form $A_i \to A_j\alpha$ by rules $A_i \to \beta_1\alpha \mid \dots \mid \beta_m\alpha$.
5. If $j = i - 1$, go to step 2., otherwise set $j := j + 1$ and go to step 4.

Greibach normal form

Theorem
Every context-free language can be generated by a grammar that does not contain left recursion. $\square$

Greibach normal form

Example
$G = (\{A, B, C\}, \{a, b\}, P, A)$, P:
$A \to BC \mid a$
$B \to CA \mid Ab$
$C \to AB \mid CC \mid a$. We apply algorithm 3.28 on this grammar.
We set $A_1 = A, A_2 = B, A_3 = C$.
Step 2: $(i = 1)$ without any changes.
Step 4: $(i = 2, j = 1)$
After substituting A we get these rules for B: $B \to CA \mid BCb \mid ab$.
Step 2: We remove left recursion at the symbol B :
$B \to CA \mid ab \mid CA\,B' \mid abB'$
$B' \to CbB' \mid Cb$
Step 4: $(i = 3, j = 1)$
$C \to BCB \mid aB \mid CC \mid a$

Greibach normal form

Example (continued)
Step 4: $(i = 3, j = 2)$
$C \to CACB \mid abCB \mid CAB'CB \mid abB'CB \mid aB \mid CC \mid a$
Step 2: $(i = 3)$
$C \to abCB \mid abB'CB \mid aB \mid a \mid abCBC' \mid abB'CBC' \mid aBC'$
$\quad \mid aC'$
$C' \to A\,CBC' \mid AB'CBC' \mid CC' \mid A\,CB \mid A\,B'CB \mid C$
Resultant grammar is $G' = (\{A, B, C, B', C'\}, \{a, b\}, P', A)$, where P' contains rules:
$A \to BC \mid a$
$B \to CA \mid ab \mid CA\,B' \mid abB'$
$B' \to CbB' \mid Cb$
$C \to abCB \mid abB'CB \mid aB \mid a \mid abCBC' \mid abB'CBC' \mid aBC'$
$\quad \mid aC'$
$C' \to ACBC' \mid AB'CBC' \mid CC' \mid ACB \mid AB'CB \mid C$