COMP4141 Theory of Computation

Lecture 7 Grammars

Ron van der Meyden

CSE, UNSW

Revision: 2016/3/24

(Credits: David Dill, Thomas Wilke, Kai Engelhardt, Peter Höfner, Rob van Glabbeek)

Grammars

CFGs are special cases of (Chomsky) Grammars.

Definition (Chomsky Grammar)

A *Chomsky Grammar* is a 4-tuple $G = (N, \Sigma, P, S)$ where N, Σ, and S are as for CFGs, but the finite set of *productions* merely satisfies

$$P \subseteq (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^* \ .$$

Write

- $\alpha \Rightarrow_G \beta$ if there exist $\alpha_1, \beta_1, \gamma, \delta \in (N \cup \Sigma)^*$ such that $\alpha = \gamma\alpha_1\delta$, $\beta = \gamma\beta_1\delta$, and $\alpha_1 \to \beta_1 \in P$.
- $\alpha \Rightarrow_G^n \beta$ if there exist $\alpha_0, \ldots, \alpha_n$ such that $\alpha_0 = \alpha$, $\alpha_n = \beta$, and $\alpha_i \Rightarrow_G \alpha_{i+1}$ for $i < n$.
- $\alpha \Rightarrow_G^* \beta$ if there exists $n \in \mathbb{N}$ such that $\alpha \Rightarrow_G^n \beta$.

$L(G) = \{ \ w \in \Sigma^* \mid S \Rightarrow_G^* w \ \}$ —the *language generated by* G.

Definition (Chomsky Hierarchy)

A grammar $G = (N, \Sigma, P, S)$ is of *type*

- **0** (or *recursively enumerable*) in the general case.
- **1** (or *context-sensitive*), if $|\alpha| \leq |\beta|$ for all productions $\alpha \to \beta \in P$, except that we allow $S \to \epsilon$ provided that also there is no occurrence of S on the RHS of any rule.
- **2** (or *context-free*), if all productions have the form $A \to \alpha$.
- **3** (or *right-linear*), if all productions are of the form $A \to w$ or $A \to wB$, where $w \in \Sigma$ and $B \in N$.

The language $L \subseteq \Sigma^*$ is said to be of *of type i* if there exists a grammar G of the respective type with $L = L(G)$.

Examples

$$G_1 : S \to \mathtt{abc} \mid \mathtt{a}A\mathtt{bc}$$
$$A\mathtt{b} \to \mathtt{b}A$$
$$A\mathtt{c} \to B\mathtt{bcc}$$
$$\mathtt{b}B \to B\mathtt{b}$$
$$\mathtt{a}B \to \mathtt{aa}A \mid \mathtt{aa}$$

$$G_2 : S \to \mathtt{0} \mid \mathtt{1} \mid (S + S) \mid (S * S)$$

$$G_3 : S \to \mathtt{a}B \mid \mathtt{b}A$$
$$A \to \mathtt{a} \mid \mathtt{a}S \mid \mathtt{b}AA$$
$$B \to \mathtt{b} \mid \mathtt{b}S \mid \mathtt{a}BB$$

$$G_4 : S \to \mathtt{a}A \mid \mathtt{a}$$
$$A \to \mathtt{a}A \mid \mathtt{a} \mid \mathtt{b}B$$
$$B \to \mathtt{a}B \mid \mathtt{b}A \mid \mathtt{b}$$

G_1 is type 1, G_2 and G_3 are type 2, and G_4 is type 3.

$S \Rightarrow_{G_1} \mathtt{a}A\mathtt{bc} \Rightarrow_{G_1} \mathtt{ab}A\mathtt{c} \Rightarrow_{G_1} \mathtt{ab}B\mathtt{bcc} \Rightarrow_{G_1} \mathtt{a}B\mathtt{bbcc} \Rightarrow_{G_1} \mathtt{aabbcc}$

$S \Rightarrow_{G_2} (S + S) \Rightarrow_{G_2} ((S * S) + S) \Rightarrow_{G_2} ((S * S) + 1) \Rightarrow_{G_2}$
$((0 * S) + 1) \Rightarrow_{G_2} ((0 * 1) + 1) \Rightarrow_{G_2}$

$S \Rightarrow_{G_3} \mathtt{a}B \Rightarrow_{G_3} \mathtt{ab}S \Rightarrow_{G_3} \mathtt{abb}A \Rightarrow_{G_3} \mathtt{abba}S \Rightarrow_{G_3} \mathtt{abbaa}B \Rightarrow_{G_3}$
$\mathtt{abbaab}$

$S \Rightarrow_{G_4} \mathtt{a}A \Rightarrow_{G_4} \mathtt{aa}A \Rightarrow_{G_4} \mathtt{aab}B \Rightarrow_{G_4} \mathtt{aabb}A \Rightarrow_{G_4} \mathtt{aabba}$

An equivalent definition for context sensitive grammars, that makes it easier to see where the name comes from:

G is context sensitive if all productions have the form $\alpha B\gamma \to \alpha\delta\gamma$, where B is a nonterminal and $\delta \neq \epsilon$, except that we allow $S \to \epsilon$, provided there is no S on the RHS of any rule.

That is, ability to rewrite B depends on the surrounding context.

Note: allowing $\delta = \epsilon$ produces a class equivalent to type 0 grammars! The nasty exception clause here and above is just to enable ϵ to be in the language in spite of the $\delta \neq \epsilon$ requirement. (Some authors just say ϵ cannot be in the language to get rid of this. Chomsky had no exception but multiple start *strings* rather than a single start symbol.)

Fundamental Questions

For grammars of type i,

- what is their expressive power?
- is there a corresponding automata model?
- are there simpler yet equally powerful subclasses, so called *normal forms*?
- can we decide:
  - the *word problem*: given G and w, is $w \in L(G)$?
  - the *emptiness problem*: given G, is $L(G) = \emptyset$?
  - the *equivalence problem*: given G_1 and G_2, is $L(G_1) = L(G_2)$?

Some Answers

Theorem

L is context-free iff $L = L(A)$ for some PDA A.

Theorem

L is right-linear iff L is regular.

Proof.

"⇒:" Let L be right-linear. We translate a right-linear grammar $G = (N, \Sigma, P, S)$ with $L = L(G)$ to an NFA with word transitions $\mathcal{A}_G = (N \cup \{\Omega\}, \Sigma, S, \delta, \{\Omega\})$ where

$$\begin{aligned} B \in \delta(A, w) \quad &\text{iff} \quad A \to wB \in P \quad \text{, and} \\ \Omega \in \delta(A, w) = \quad &\text{iff} \quad A \to w \in P \quad . \end{aligned}$$

It follows that $L(\mathcal{A}_G) = L(G)$. □

Proof.

"$\Leftarrow$:" Let L be regular. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with $L = L(M)$ and, w.l.o.g., no transition to q_0. Define $G_M = (Q, \Sigma, P, q_0)$ by

$$\begin{aligned} A \to aB \in P \quad &\text{iff} \quad B \in \delta(A, a) \ , \\ A \to a \in P \quad &\text{iff} \quad \delta(A, a) \cap F \neq \emptyset \ , \text{ and} \\ q_0 \to \epsilon \in P \quad &\text{iff} \quad q_0 \in F \ . \end{aligned}$$

It follows that $L(G_M) = L(M)$. $\square$

Deciding emptiness of a regular language

How to decide emptiness of a regular language L depends on its representation, of which we've met a few.

NFA: when given as $L(A)$ of an NFA $(Q, \Sigma, \delta, q_0, F)$ (or DFA, ϵ-NFA) is an exercise in graph reachability: Is there a final state that can be reached from the initial state?

This can be done by a depth-first search in time linear in the number of edges and vertices. (But note there could be $|Q|^2$ edges.)

Deciding emptiness of a regular language (cont.)

$\mathbf{RE}_\Sigma$: When L is given as $L(R)$ of a regular expression R then we can *abstract* inductively as follows:

Base: $L(\emptyset) = \emptyset$, $L(\epsilon) \neq \emptyset$, and $L(a) \neq \emptyset$.

Induction:

$L(R_1 \cup R_2) = \emptyset$ iff $L(R_1) = L(R_2) = \emptyset$.

$L(R_1 \circ R_2) = \emptyset$ iff $L(R_1) = \emptyset$ or $L(R_2) = \emptyset$.

$L(R_1^*) \neq \emptyset$.

$L((R_1)) = \emptyset$ iff $L(R_1) = \emptyset$.

This implies that $L(R) = \emptyset$ can be decided in $O(|R|)$ time.

The word problem for regular languages

DFA: easy, just feed the word to the automaton.

NFA: marking algorithm: on input $w = a_1 \dots a_{|w|}$

1. Set of marks $M := \{q_0\}$.
2. For $i = 1$ to $|w|$ do
   $M := \bigcup_{q \in M} \delta(q, a_i)$
3. Return whether $F \cap M \neq \emptyset$.

Others: translate to an equivalent DFA. See above.

The equivalence problem for regular languages

DFAs: for DFAs A_1 and A_2 construct an DFA recognising the symmetric set difference,

$$(L(A_1) \cap \overline{L(A_2)}) \cup (L(A_2) \cap \overline{L(A_1)})$$

by employing the standard constructions for complement and union. Then check emptiness.

Others: translate to equivalent DFAs. See above.

Emptiness of CFLs

Given a CFG $G = (V, \Sigma, P, S)$ the emptiness problem can be decided as follows:

1. Mark the terminals and ϵ, as *generating*
2. Mark as generating all those non-terminals that have a production with only generating symbols in the RHS.
3. Repeat step 2 until nothing new is marked generating.
4. Check whether the start symbol is marked as generating.

Chomsky normal form

For many of the remaining questions, it is convenient to introduce a particularly simple class of CFGs that is still as powerful as CFGs in general.

Definition

A context free (type 2) grammar is in *Chomsky normal form* if every production is of one of the forms

- $S \to \epsilon$, or
- $A \to BC$ where B and C are not S, or
- $A \to a$.

Theorem

Any CFL is generated by a CFG in Chomsky normal form.

Proof.

Let $G = (V, \Sigma, P, S)$ be a CFG.

Step 0: Define an equivalent CFG $G_0 = (V \cup \{S_0\}, \Sigma, P_0, S_0)$ with a fresh start variable S_0 and the production $S_0 \to S$. We also remove all productions of the form $A \to \epsilon$ and patch this up by introducing new productions for every occurrence of A in a body with that occurrence removed.

(E.g., $A \to \epsilon \mid \mathtt{a}A\mathtt{b}A$ becomes $A \to \mathtt{ab} \mid \mathtt{ab}A \mid \mathtt{a}A\mathtt{b} \mid \mathtt{a}A\mathtt{b}A$.)

Productions $B \to A$ are replaced by $B \to \epsilon$ unless that is one we had removed earlier.

Repeat until ϵ occurs at most in $S_0 \to \epsilon$.

Step 1: Define an equivalent CFG $G_1 = (V_1, \Sigma, P_1, S_0)$ with fresh non-terminals for terminals $V_1 = V \cup \{S_0\} \cup \{\ X_a \mid a \in \Sigma\ \}$. Here P_1 is derived from P_0 by replacing all occurrences of a terminal $a \in \Sigma$ in the body of a production by the corresponding non-terminal X_a and then adding productions $X_a \to a$.

Proof cont.

Step 2: Define an equivalent CFG $G_2 = (V_1, \Sigma, P_2, S_0)$. To generate P_2 from P_1, eliminate productions of the form $A \rightarrow B$ by

1. determining all derivations $A_1 \Rightarrow_{G_1} \ldots \Rightarrow_{G_1} A_k \Rightarrow_{G_1} \alpha \notin V_1$ not containing repetitions of non-terminals,
2. drop all productions $A \rightarrow B$, and
3. introduce $A_1 \rightarrow \alpha$ for each of the derivations determined before.

Step 3: Define an equivalent CFG $G_3 = (V_3, \Sigma, P_3, S_0)$. To generate P_3 from P_2, replace all productions $A \to B_1 \ldots B_n$ with $n > 2$ by $A \to B_1 C_1$, $C_1 \to B_2 C_2$, ..., $C_{n-2} \to B_{n-1} B_n$ for fresh non-terminals C_i. (These are distinct for each such production.) $\square$

The word problem for CFLs

The word problem for CFGs is decidable.
More precisely we'll show that

Theorem

Let $G = (V, \Sigma, P, S)$ be a CFG in Chomsky normal form. Then the CYK-algorithm decides $w \in L(G)$ in time $O(|w|^3)$ for $w \in \Sigma^*$.

The CYK-algorithm (for Cocke-Younger-Kasami) works as follows.
Given $w = a_1 \dots a_n$ compute for $i, j \in \{1, \dots, n\}$ the set $N_{ij} = \{ A \in V \mid A \Rightarrow_G^* a_i \dots a_j \}$ of non-terminals generating $a_i \dots a_j$. Then

$$w \in L(G) \quad \text{iff} \quad S \in V_{1n} \ .$$

The details are not in [Sipser2006].

CYK-Algorithm

for CFG $G = (V, \Sigma, P, S)$ in Chomsky normal form.
"On input $w = a_1 \dots a_n$

1. for $i, j := 1$ to n do

$$N_{i,j} := \begin{cases} \{\ A \in V \mid A \to a_i \in P\ \} & \text{if } i = j \\ \emptyset & \text{otherwise} \end{cases}$$

2. for $d := 1$ to $n - 1$ and $i := 1$ to $n - d$ do
   1. $j := i + d$
   2. for $k := i$ to $j - 1$ do

$$N_{i,j} := N_{i,j} \cup \left\{ A \in V \;\middle|\; \exists B, C \left(\begin{array}{l} A \to BC \in P \wedge \\ B \in N_{i,k} \wedge C \in N_{k+1,j} \end{array} \right) \right\}$$

3. Return whether $S \in N_{1,n}$.

Preview of undecidable CFL problems

Later we'll develop a theory that allows us to prove rigorously that there are problems that *cannot be solved by any algorithm* that can be implemented as a computer program.

Such problems are called *undecidable*.

Some simple decision problems in the realm of CFLs are undecidable:

- Is a given CFG ambiguous?
- Is any CFG for a given CFL necessarily ambiguous?
- Is the intersection of two given CFLs empty?
- Are two given CFGs/PDAs equivalent?
- Does a given CFG generate all strings Σ^*?

—THE END—