Finite Automata Theory and Formal Languages

TMV027/DIT321– LP4 2015

Lecture 13
Ana Bove

May 18th 2015

Overview of today's lecture:

- Closure properties of CFL;
- Decision properties of CFL;
- Push-down automata.

Recap: Context-free Grammars

- Regular languages are also context-free;
- Chomsky hierarchy;
- Simplification of grammars:
  - Elimination of ϵ-productions;
  - Elimination of unit productions;
  - Elimination of useless symbols:
    - Elimination of non-generating symbols;
    - Elimination of non-reachable symbols;
- Chomsky normal forms;
- Pumping lemma for context-free languages.

Closure under Union

Theorem: *Let $G_1 = (V_1, T, \mathcal{R}_1, S_1)$ and $G_2 = (V_2, T, \mathcal{R}_2, S_2)$ be CFG. Then $\mathcal{L}(G_1) \cup \mathcal{L}(G_2)$ is a context-free language.*

Proof: Let us assume $V_1 \cap V_2 = \emptyset$ (easy to get via renaming).

Let S be a fresh variable.

We construct $G = (V_1 \cup V_2 \cup \{S\}, T, \mathcal{R}_1 \cup \mathcal{R}_2 \cup \{S \to S_1 \mid S_2\}, S)$.

It is now easy to see that $\mathcal{L}(G) = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$ since a derivation will have the form

$$S \Rightarrow S_1 \Rightarrow^* w \text{ if } w \in \mathcal{L}(G_1)$$

or

$$S \Rightarrow S_2 \Rightarrow^* w \text{ if } w \in \mathcal{L}(G_2)$$

Closure under Concatenation

Theorem: *Let $G_1 = (V_1, T, \mathcal{R}_1, S_1)$ and $G_2 = (V_2, T, \mathcal{R}_2, S_2)$ be CFG. Then $\mathcal{L}(G_1)\mathcal{L}(G_2)$ is a context-free language.*

Proof: Again, let us assume $V_1 \cap V_2 = \emptyset$.

Let S be a fresh variable.

We construct $G = (V_1 \cup V_2 \cup \{S\}, T, \mathcal{R}_1 \cup \mathcal{R}_2 \cup \{S \to S_1 S_2\}, S)$.

It is now easy to see that $\mathcal{L}(G) = \mathcal{L}(G_1)\mathcal{L}(G_2)$ since a derivation will have the form

$$S \Rightarrow S_1 S_2 \Rightarrow^* uv$$

with

$$S_1 \Rightarrow^* u \text{ and } S_2 \Rightarrow^* v$$

for $u \in \mathcal{L}(G_1)$ and $v \in \mathcal{L}(G_2)$.

Closure under Closure

Theorem: *Let $G = (V, T, \mathcal{R}, S)$ be a CFG. Then $\mathcal{L}(G)^+$ and $\mathcal{L}(G)^*$ are context-free languages.*

Proof: Let S' be a fresh variable.

We construct $G+ = (V \cup \{S'\}, T, \mathcal{R} \cup \{S' \to S \mid SS'\}, S')$ and $G* = (V \cup \{S'\}, T, \mathcal{R} \cup \{S' \to \epsilon \mid SS'\}, S')$.

It is easy to see that $S' \Rightarrow \epsilon$ in $G*$.

It is also easy to see that $S' \Rightarrow^* S \Rightarrow^* w$ if $w \in \mathcal{L}(G)$ is a valid derivation both in $G+$ and in $G*$.

In addition, if $w_1, \ldots, w_k \in \mathcal{L}(G)$, it is easy to see that the derivation

$$\begin{aligned} S' &\Rightarrow SS' \Rightarrow^* w_1 S' \Rightarrow w_1 SS' \Rightarrow^* w_1 w_2 S' \Rightarrow^* \ldots \\ &\Rightarrow^* w_1 w_2 \ldots w_{k-1} S' \Rightarrow^* w_1 w_2 \ldots w_{k-1} S \Rightarrow^* w_1 w_2 \ldots w_{k-1} w_k \end{aligned}$$

is a valid derivation both in $G+$ and in $G*$.

Non Closure under Intersection

Example: Consider the following languages over $\{a, b, c\}$:

$$\mathcal{L}_1 = \{a^k b^k c^m \mid k, m > 0\}$$

$$\mathcal{L}_2 = \{a^m b^k c^k \mid k, m > 0\}$$

It is easy to give CFG generating both $\mathcal{L}_1$ and $\mathcal{L}_2$, hence $\mathcal{L}_1$ and $\mathcal{L}_2$ are CFL.

However $\mathcal{L}_1 \cap \mathcal{L}_2 = \{a^k b^k c^k \mid k > 0\}$ is not a CFL (see slide 26 lecture 12).

Closure under Intersection with Regular Language

Theorem: *If $\mathcal{L}$ is a CFL and $\mathcal{P}$ is a RL then $\mathcal{L} \cap \mathcal{P}$ is a CFL.*

Proof: See Theorem 7.27 in the book.

(It uses *push-down automata* which we have not seen.)

Example: Consider the following language over $\Sigma = \{0, 1\}$:

$$\mathcal{L} = \{ww \mid w \in \Sigma^*\}$$

Consider now $\mathcal{L}' = \mathcal{L} \cap \mathcal{L}(0^*1^*0^*1^*) = \{0^n1^m0^n1^m \mid n, m \geqslant 0\}$.

$\mathcal{L}'$ is not a CFL (see additional exercise 4 for week 7).

Hence $\mathcal{L}$ cannot be a CFL since $\mathcal{L}(0^*1^*0^*1^*)$ is a RL.

Non Closure under Complement

Theorem: *CFL are not closed under complement.*

Proof: Notice that

$$\mathcal{L}_1 \cap \mathcal{L}_2 = \overline{\overline{\mathcal{L}_1} \cup \overline{\mathcal{L}_2}}$$

If CFL are closed under complement then they should be closed under intersection (since they are closed under union).

Then CFL are in general not closed under complement.

Closure under Difference?

Theorem: *CFL are not closed under difference.*

Proof: Let $\mathcal{L}$ be a CFL over Σ.

It is easy to give a CFG that generates Σ^*.

Observe that $\overline{\mathcal{L}} = \Sigma^* - \mathcal{L}$.

Then if CFL are closed under difference they would also be closed under complement.

Theorem: *If $\mathcal{L}$ is a CFL and $\mathcal{P}$ is a RL then $\mathcal{L} - \mathcal{P}$ is a CFL.*

Proof: Observe that $\overline{\mathcal{P}}$ is a RL and $\mathcal{L} - \mathcal{P} = \mathcal{L} \cap \overline{\mathcal{P}}$.

Closure under Reversal and Prefix

Theorem: *If $\mathcal{L}$ is a CFL then so is $\mathcal{L}^{\mathrm{r}} = \{\mathrm{rev}(w) \mid w \in \mathcal{L}\}$.*

Proof: Given a CFG $G = (V, T, \mathcal{R}, S)$ for $\mathcal{L}$ we construct the grammar $G^{\mathrm{r}} = (V, T, \mathcal{R}^{\mathrm{r}}, S)$ where $\mathcal{R}^{\mathrm{r}}$ is such that, for each rule $A \to \alpha$ in $\mathcal{R}$, then $A \to \mathrm{rev}(\alpha)$ is in $\mathcal{R}^{\mathrm{r}}$.

One should show by induction on the length of the derivations in G and G^{r} that $\mathcal{L}(G^{\mathrm{r}}) = \mathcal{L}^{\mathrm{r}}$.

Theorem: *If $\mathcal{L}$ is a CFL then so is* $\mathrm{Prefix}(\mathcal{L})$.

Proof: For closure under prefix see exercise 7.3.1 part a) in the book.

Closure under Homomorphisms

Theorem: *CFL are closed under homomorphisms.*

Proof: See Theorem 7.24 point 4 in the book.

(It uses the notion of *substitution* which we have not seen.)

Decision Properties of Context-Free Languages

Very little can be answered when it comes to CFL.

The major tests we can answer are whether:

- The language is empty;

  (See the algorithm that tests for generating symbols in slide 6 lecture 12:
  if $\mathcal{L}$ is a CFL given by a grammar with start variable S, then $\mathcal{L}$ is empty if S is not generating.)

- A certain string belongs to the language.

Testing Membership in a Context-Free Language

Checking if $w \in \mathcal{L}(G)$, where $|w| = n$, by trying all productions may be exponential on n.

An efficient way to check for membership in a CFL is based on the idea of *dynamic programming*.

(Method for solving complex problems by breaking them down into simpler problems, applicable mainly to problems where many of their subproblems are really the same; not to be confused with the *divide and conquer* strategy.)

The algorithm is called the *CYK algorithm* after the 3 people who independently discovered the idea: Cock, Younger and Kasami.

It is a $O(n^3)$ algorithm.

The CYK Algorithm

Let $G = (V, T, \mathcal{R}, S)$ be a CFG in CNF and $w = a_1 a_2 \ldots a_n \in T^*$.

Does $w \in \mathcal{L}(G)$?

In the CYK algorithm we fill a table

$$
\begin{array}{|cccccc}
V_{1n} & & & & & \\
V_{1(n-1)} & V_{2n} & & & & \\
\vdots & \vdots & & & & \\
V_{12} & V_{23} & V_{34} & \ldots & V_{(n-1)n} & \\
V_{11} & V_{22} & V_{33} & \ldots & V_{(n-1)(n-1)} & V_{nn} \\
\hline
a_1 & a_2 & a_3 & \ldots & a_{n-1} & a_n
\end{array}
$$

where $V_{ij} \subseteq V$ is the set of A's such that $A \Rightarrow^* a_i a_{i+1} \ldots a_j$.

We want to know if $S \in V_{1n}$, hence $S \Rightarrow^* a_1 a_2 \ldots a_n$.

CYK Algorithm: Observations

- Each row corresponds to the substrings of a certain length:
  - bottom row is length 1,
  - second from bottom is length 2,
  - ...
  - top row is length n;
- We work row by row upwards and compute the V_{ij}'s;
- In the bottom row we have $i = j$, that is, ways of generating the string a_i;
- V_{ij} is the set of variables generating $a_i a_{i+1} \ldots a_j$ of length $j - i + 1$ (hence, V_{ij} is in row $j - i + 1$);
- In the rows below that of V_{ij} we have all ways to generate shorter strings, including all prefixes and suffixes of $a_i a_{i+1} \ldots a_j$.

CYK Algorithm: Table Filling

Remember we work with a CFG in CNF.

We compute V_{ij} as follows:

Base case: First row in the table. Here $i = j$.
Then $V_{ii} = \{A \mid A \to a_i \in \mathcal{R}\}$.

Induction step: To compute V_{ij} for $i < j$ we have all V_{pq}'s in rows below.

The length of the string is at least 2, so $A \Rightarrow^* a_i a_{i+1} \ldots a_j$ starts with $A \Rightarrow BC$ such that $B \Rightarrow^* a_i a_{i+1} \ldots a_k$ and $C \Rightarrow^* a_{k+1} \ldots a_j$ for some k.

So $A \in V_{ij}$ if $\exists k, i \leqslant k < j$ such that

- $B \in V_{ik}$ and $C \in V_{(k+1)j}$;
- $A \to BC \in \mathcal{R}$.

We need to look at
$(V_{ii}, V_{(i+1)j}), (V_{i(i+1)}, V_{(i+2)j}), \ldots, (V_{i(j-1)}, V_{jj})$.

CYK Algorithm: Example

Consider the grammar given by the rules

$$S \to AB \mid BA \qquad A \to AS \mid a \qquad B \to BS \mid b$$

and starting symbol S.

Does *abba* belong to the language generated by the grammar?

We fill the corresponding table:

$\{S\}$			
$\emptyset$	$\{B\}$		
$\{S\}$	$\emptyset$	$\{S\}$	
$\{A\}$	$\{B\}$	$\{B\}$	$\{A\}$
a	b	b	a

$S \in V_{14}$ then $S \Rightarrow^* abba$.

CYK Algorithm: Example

Consider the grammar given by the rules

$$\begin{array}{ll} S \to XY & X \to XA \mid a \mid b \\ Y \to AY \mid a & A \to a \end{array}$$

and starting symbol S.

Does *babaa* belong to the language generated by the grammar?

We fill the corresponding table:

$\emptyset$				
$\emptyset$	$\emptyset$			
$\emptyset$	$\emptyset$	$\{S, X\}$		
$\{S, X\}$	$\emptyset$	$\{S, X\}$	$\{S, X, Y\}$	
$\{X\}$	$\{A, X, Y\}$	$\{X\}$	$\{A, X, Y\}$	$\{A, X, Y\}$
b	a	b	a	a

$S \notin V_{15}$ then $S \not\Rightarrow^* babaa$.

Undecidable Problems for Context-Free Grammars/Languages

Definition: An *undecidable problem* is a decision problem for which it is impossible to construct a single algorithm that always leads to a correct yes-or-no answer.

Example: Halting problem: does this program terminate?

The following problems are undecidable:

- Is the CFG G ambiguous?
- Is the CFL $\mathcal{L}$ inherently ambiguous?
- If $\mathcal{L}_1$ and $\mathcal{L}_2$ are CFL, is $\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset$?
- If $\mathcal{L}_1$ and $\mathcal{L}_2$ are CFL, is $\mathcal{L}_1 = \mathcal{L}_2$? is $\mathcal{L}_1 \subseteq \mathcal{L}_2$?
- If $\mathcal{L}$ is a CFL and $\mathcal{P}$ a RL, is $\mathcal{P} = \mathcal{L}$? is $\mathcal{P} \subseteq \mathcal{L}$?
- If $\mathcal{L}$ is a CFL over Σ, is $\mathcal{L} = \Sigma^*$?

Push-down Automata

Push-down automata (PDA) are essentially ϵ-NFA with the addition of a *stack* where to store information.

The stack is needed to give the automata extra "memory".

Observe we can only access the last element that was added to the stack!

Example: To recognise the language 0^n1^n we proceed as follows:

- When reading the 0's, we push a symbol into the stack;
- When reading the 1's, we pop the symbol on top of the stack;
- We accept the word if when we finish reading the input then the stack is empty.

The languages accepted by the PDA are exactly the CFL.
See the book, sections 6.1–6.3.

Variation of Push-down Automata

DPDA = DFA + stack: Accepts a language that is between RL and CFL.
The lang. accepted by DPDA have unambiguous grammars.
However, not all languages that have unambiguous grammars can be accepted by these DPDA.

Example: The language generated by the unambiguous grammar

$$S \to 0S0 \mid 1S1 \mid \epsilon$$

cannot be recognised by a DPDA.
See section 6.4 in the book.

2 or more stacks: A PDA with at least 2 stacks is as powerful as a TM.
Hence these PDA can recognise the *recursively enumerable* languages (more on this later).
See section 8.5.2.

Overview of Next Lecture

Section 8:

- Turing machines.

Guest lecture by Prof. *Aarne Ranta*

Automata and Grammars in
Programming Language Technology