Finite Automata Theory and Formal Languages

TMV027/DIT321– LP4 2013

Lecture 10
Ana Bove

April 30th 2013

Overview of today's lecture:

- Context-free grammars;
- Derivations:
- Parse trees.

Context-Free Grammars

Not all languages are regular languages as we have seen.

Context-free grammars (CFG) define the so called *context-free languages*.

We have that (we will see this next lecture)

$$\textit{regular languages} \subset \textit{context-free languages}$$

CFG play an important role in the description and design of programming languages and compilers, for example, for the implementation of parsers.

They have been developed in the mid-1950s by Noam Chomsky.

Example: Palindromes

Let us consider the language of *palindromes* over $\Sigma = \{0, 1\}$.

That is, $\mathcal{L} = \{w \in \Sigma^* \mid w = \text{rev}(w)\}$.

Example of palindromes over Σ are ϵ, 0110, 00011000, 010.

We can use the Pumping Lemma for RL to show that $\mathcal{L}$ is not regular.

How can $\mathcal{L}$ be defined?

We have that (inductive definition):

- ϵ, 0 and 1 are in $\mathcal{L}$;
- if $w \in \mathcal{L}$ then $0w0$ and $1w1$ are also in $\mathcal{L}$.

Example: Palindromes

The definition of palindromes as a CFG is the following

$$\begin{array}{lcl} P & \rightarrow & \epsilon \\ P & \rightarrow & 0 \\ P & \rightarrow & 1 \\ P & \rightarrow & 0P0 \\ P & \rightarrow & 1P1 \end{array}$$

The variable P represents the class of the strings that are palindromes.

The rules say how to construct the strings in the language.

Example: Simple Expressions

Here we define 2 classes of strings: those representing simple numerical expressions and those representing Boolean expressions.

$$
\begin{array}{lcl}
E & \rightarrow & 0 \\
E & \rightarrow & 1 \\
E & \rightarrow & E + E \\
E & \rightarrow & \text{if } B \text{ then } E \text{ else } E \\
B & \rightarrow & \text{True} \\
B & \rightarrow & \text{False} \\
B & \rightarrow & E < E \\
B & \rightarrow & E == E
\end{array}
$$

Compact Notation

We can group all productions defining elements in a certain class to have a more compact notation.

Example: Palindromes can be defined as

$$P \rightarrow \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$$

Example: The expressions can be defined as

$$
\begin{array}{lcl}
E & \rightarrow & 0 \mid 1 \mid E + E \mid \text{if } B \text{ then } E \text{ else } E \\
B & \rightarrow & \text{True} \mid \text{False} \mid E < E \mid E == E
\end{array}
$$

Context-Free Grammars

Definition: A *context-free grammar* is a 4-tuple $G = (V, T, \mathcal{R}, S)$ where:

- V is a finite set of *variables* or *non-terminals*: each variable represents a language or set of strings;
- T is a finite set of *symbols* or *terminals*: think of T as the alphabet of the language you are defining;
- $\mathcal{R}$ is a finite set of *rules* or *productions* which recursively define the language. Each production consists of:
  - A variable being defined in the production;
  - The symbol "$\rightarrow$";
  - A string of 0 or more terminals and variables called the *body* of the production;
- S is the *start variable* and represents the language we are defining; other variables define the auxiliary classes needed to define our language.

Notation for Context-Free Grammars

We use the following convention when talking about CFG:

- $a, b, c, \ldots, 0, 1, \ldots, (,), +, \%, \ldots$ are terminal symbols;
- $A, B, C, \ldots$ are variables (non-terminals);
- $w, x, y, z, \ldots$ are strings of terminals;
- $X, Y, \ldots$ are either a terminal or a variable;
- $\alpha, \beta, \gamma, \ldots$ are strings of terminals and/or variables;
  In particular, they can also represent strings with *only* variables.

Derivation Using Grammars

We use the productions of a CFG to infer that a string w is in the language of given variable.

Example: Let us derive that 0110110 is a palindrome.

We can do this in 2 (equivalent) ways:

Recursive inference: From body to head.

1)	0	is palindrome by rule	$P \to 0$
2)	101	is palindrome using	1) and rule $P \to 1P1$
3)	11011	is palindrome using	2) and rule $P \to 1P1$
4)	0110110	is palindrome using	3) and rule $P \to 0P0$

Derivation: (Notation: $\Rightarrow$) From head to body.

$$P \Rightarrow 0P0 \Rightarrow 01P10 \Rightarrow 011P110 \Rightarrow 0110110$$

Formal Definition of Derivation

Let $G = (V, T, \mathcal{R}, S)$ be a CFG.

Definition: Let $A \in V$ and $\alpha, \beta \in (V \cup T)^*$. Let $A \to \gamma \in \mathcal{R}$. Then $\alpha A \beta \Rightarrow \alpha\gamma\beta$ is one *derivation step* (alternative $\alpha A \beta \overset{G}{\Rightarrow} \alpha\gamma\beta$).

Example:

$$B \Rightarrow E == E \Rightarrow 0 == E \Rightarrow 0 == E + E \Rightarrow 0 == E + 1 \Rightarrow 0 == 0 + 1$$

We can define a relation performing zero or more derivation steps.

Definition: The *reflexive-transitive closure* of $\Rightarrow$ is the relation $\Rightarrow^*$ (alternative $\overset{G}{\Rightarrow}{}^*$) defined as follows:

$$\frac{}{\alpha \Rightarrow^* \alpha} \qquad \frac{\alpha \Rightarrow^* \beta \qquad \beta \Rightarrow \gamma}{\alpha \Rightarrow^* \gamma}$$

Example: $B \Rightarrow^* 0 == 0 + 1$.

Leftmost and Rightmost Derivations

At every derivation step we can choose to replace *any* variable by the right-hand side of one of its productions.

Two particular derivations are:

Leftmost derivation: Notations: $\overset{lm}{\Rightarrow}$ and $\overset{lm}{\Rightarrow}^*$.
At each step we choose to replace the *leftmost variable*.

Example: $B \overset{lm}{\Rightarrow}^* 0 == 0 + 1$

$$B \overset{lm}{\Rightarrow} E == E \overset{lm}{\Rightarrow} 0 == E \overset{lm}{\Rightarrow} 0 == E + E \overset{lm}{\Rightarrow} 0 == 0 + E \overset{lm}{\Rightarrow} 0 == 0 + 1$$

Rightmost derivation: Notations: $\overset{rm}{\Rightarrow}$ and $\overset{rm}{\Rightarrow}^*$.
At each step we choose to replace the *rightmost variable*.

Example: $B \overset{rm}{\Rightarrow}^* 0 == 0 + 1$

$$B \overset{rm}{\Rightarrow} E == E \overset{rm}{\Rightarrow} E == E+E \overset{rm}{\Rightarrow} E == E+1 \overset{rm}{\Rightarrow} E == 0+1 \overset{rm}{\Rightarrow} 0 == 0+1$$

Observations on Derivations

Observe: If we have $A \Rightarrow^* \gamma$ then we also have $\alpha A \beta \Rightarrow^* \alpha\gamma\beta$.

The same sequence of steps that took us from A to γ will also take us from $\alpha A \beta$ to $\alpha\gamma\beta$.

Example: We have $E \Rightarrow^* 0 + 1$ since $E \Rightarrow E + E \Rightarrow 0 + E \Rightarrow 0 + 1$.

This same derivation justifies $E == E \Rightarrow^* E == 0 + 1$ since

$$E == E \Rightarrow E == E + E \Rightarrow E == 0 + E \Rightarrow E == 0 + 1$$

Observations on Derivations

Observe: If we have $A \Rightarrow X_1 X_2 \dots X_k \Rightarrow^* w$, then we can break w into pieces $w_1, w_2, \dots, w_k$ such that $X_i \Rightarrow^* w_i$. If X_i is a terminal then $X_i = w_i$.

This can be showed by proving (by induction on the length of the derivation) that if $X_1 X_2 \dots X_k \Rightarrow^* \alpha$ then all positions of α that come from expansion of X_i are to the left of all positions that come from the expansion of X_j if $i < j$.

To obtain $X_i \Rightarrow^* w_i$ from $A \Rightarrow^* w$ we need to remove all positions of the sentential form that are to the left and to the right of all the positions derived from X_i, and all steps not relevant for the derivation of w_i from X_i.

Example: Let $B \Rightarrow E == E \Rightarrow E == E + E \Rightarrow E == E + 0 \Rightarrow E == E + E + 0 \Rightarrow E == 0 + E + 0 \Rightarrow 1 == 0 + E + 0 \Rightarrow 1 == 0 + 1 + 0$.

The derivation from the middle E in the sentential form $E == E + E$ is $E \Rightarrow E + E \Rightarrow 0 + E \Rightarrow 0 + 1$.

Sentential Forms

Derivations from the start variable have an special role.

Definition: Let $G = (V, T, \mathcal{R}, S)$ be a CFG and let $\alpha \in (V \cup T)^*$. Then $S \Rightarrow^* \alpha$ is called a *sentential form*.

(We say *left sentential form* if $S \overset{lm}{\Rightarrow}{}^* \alpha$ or *right sentential form* if $S \overset{rm}{\Rightarrow}{}^* \alpha$.)

Example: A sentential form: $B \Rightarrow^* 0 == E + 1$ since

$$B \Rightarrow E == E \Rightarrow 0 == E \Rightarrow 0 == E + E \Rightarrow 0 == E + 1$$

A left sentential form: $B \overset{lm}{\Rightarrow}{}^* E == 0 + 1$ since

$$B \overset{lm}{\Rightarrow} E == E \overset{lm}{\Rightarrow} 0 == E \overset{lm}{\Rightarrow} 0 == E + E \overset{lm}{\Rightarrow} 0 == 0 + E$$

A right sentential form: $B \overset{rm}{\Rightarrow}{}^* E == 0 + 1$ since

$$B \overset{rm}{\Rightarrow} E == E \overset{rm}{\Rightarrow} E == E + E \overset{rm}{\Rightarrow} E == E + 1 \overset{rm}{\Rightarrow} E == 0 + 1$$

Language of a Grammar

Definition: Let $G = (V, T, \mathcal{R}, S)$ be a CFG. The language of G, denoted $\mathcal{L}(G)$ is the set of terminal strings that can be derived from the start variable, that is,

$$\mathcal{L}(G) = \{w \in T^* \mid S \overset{G}{\Rightarrow}^* w\}$$

Definition: If a language $\mathcal{L}$ is the language of a certain context-free grammar, then $\mathcal{L}$ is said to be a *context-free language*.

Example: C++ Compound Statements

A context free grammar for statements:

$$\begin{array}{lcl}
S & \rightarrow & \{LC\} \\
LC & \rightarrow & C\ LC \mid \epsilon \\
C & \rightarrow & S \mid \mathit{if}\ (E)\ C \mid \mathit{if}\ (E)\ C\ \mathit{else}\ C \mid \\
 & & \mathit{while}\ (E)\ C \mid \mathit{do}\ C\ \mathit{while}\ (E) \mid \mathit{for}\ (C\ E; E)\ C \mid \\
 & & \mathit{case}\ E : C \mid \mathit{switch}\ (E)\ C \mid \mathit{return}\ E; \mid \mathit{goto}\ \mathit{Id}; \\
 & & \mathit{break}; \mid \mathit{continue}; \\
E & \rightarrow & \ldots \\
\mathit{Id} & \rightarrow & L\ \mathit{LLoD} \\
\mathit{LLoD} & \rightarrow & L\ \mathit{LLoD} \mid D\ \mathit{LLoD} \mid \epsilon \\
L & \rightarrow & A \mid B \mid \ldots \mid Z \mid a \mid b \mid \ldots \mid z \\
D & \rightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}$$

A regular expression for identifiers:

$$(A \mid \ldots \mid Z \mid a \mid \ldots \mid z)(A \mid \ldots \mid Z \mid a \mid \ldots \mid z \mid 0 \mid \ldots \mid 9)^*$$

Examples of Context Free Grammars

Exercise: Construct a CFG generating $\{0^i 1^j \mid j \geqslant i \geqslant 0\}$.

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

Exercise: Construct a grammar for the following language:

$$\{a^n b^n c^m d^m \mid n, m \geqslant 1\} \cup \{a^n b^m c^m d^n \mid n, m \geqslant 1\}$$

$$\begin{array}{lcl} S & \to & AB \mid C \\ A & \to & aAb \mid ab \\ B & \to & cBd \mid cd \\ C & \to & aCd \mid aDd \\ D & \to & bDc \mid bc \end{array}$$

Context-Free Grammars and Inductive Definitions

Each CFG can be seen as an inductive definition.

Example: Consider the grammar for palindromes in slide 3 (and 5).

It can be seen as the following definition:

Base Cases:
- The empty string is a palindrome;
- 0 and 1 are palindromes.

Inductive Steps: If w is a palindrome, then so are $0w0$ and $1w1$.

A natural way then to do proofs on context-free languages is to follow this inductive structure.

Proofs About a Grammar

When we want to prove something about a grammar we usually need to prove an statement for each of the variables in the grammar.

(Compare this with proofs about FA where we needed statements for each state.)

Proofs about grammars are in general done:

- by induction on the length of a certain string of the language;
- by induction on the length of a derivation of a certain string;
- by induction on the structure of the strings in the language;
- by induction on the height of the parse tree (we shall see this later).

Example: Proof About a Grammar

Lemma: *Let G be the grammar presented on slide 3. Then $\mathcal{L}(G)$ is the set of palindromes over $\{0,1\}$.*

Proof: We will prove that if $w \in \{0,1\}^*$, then $w \in \mathcal{L}(G)$ iff $w = \text{rev}(w)$.

If) Let w be a palindrome.
We prove by course of value induction on $|w|$ that $w \in \mathcal{L}(G)$.

Base cases: If $|w| = 0$ or $|w| = 1$ then w is ϵ, 0 or 1.
We have productions $P \to \epsilon$, $P \to 0$ and $P \to 1$.
Then $P \Rightarrow^* w$ so $w \in \mathcal{L}(G)$.

Inductive Steps: Assume $|w| > 1$.
Since $w = \text{rev}(w)$ then $w = 0w'0$ or $w = 1w'1$, and $w' = \text{rev}(w')$.
By inductive hypothesis then $P \Rightarrow^* w'$.
If $w = 0w'0$ we have $P \Rightarrow 0P0 \Rightarrow^* 0w'0 = w$ so $w \in \mathcal{L}(G)$.
Similarly if $w = 1w'1$.

Example: Proof About a Grammar (Cont.)

Only-if) Let $w \in \mathcal{L}(G)$, that is $P \Rightarrow^* w$.

We prove by induction on the number of steps in the derivation of w that $w = \text{rev}(w)$.

Base case: If the derivation is in one step then we should have $P \Rightarrow \epsilon$, $P \Rightarrow 0$ and $P \Rightarrow 1$. In all cases w is a palindrome.

Inductive Step: Our IH is that if $P \Rightarrow^* w'$ in $n > 0$ steps then w' is a palindrome.

Suppose we have a derivation with $n + 1$ steps.

Then it must be of the form $P \Rightarrow 0P0 \Rightarrow^* 0w'0 = w$ or $P \Rightarrow 1P1 \Rightarrow^* 1w'1 = w$.

Observe that we should have that $P \Rightarrow^* w'$ in $n > 0$ steps.

Hence by IH w' is a palindrome and then so is w.

Example: Proof About a Grammar

Exercise: Consider the grammar $S \rightarrow \epsilon \mid 0S1 \mid S1$.
Show that if $S \Rightarrow^* w$ then $\#_1(w) \geqslant \#_0(w)$.

Proof: By induction on the length of the derivation.

If length is 1 then it should be $S \Rightarrow \epsilon$ and $\#_1(\epsilon) \geqslant \#_0(\epsilon)$ holds.

Let us assume $S \Rightarrow^* w'$ in $n \geqslant 1$ steps. By IH $\#_1(w') \geqslant \#_0(w')$.

Let $S \Rightarrow^* w$ in $n + 1$ steps, with $n \geqslant 1$. Then we have:

- $S \Rightarrow 0S1 \Rightarrow^n 0w'1$: By IH $\#_1(w') \geqslant \#_0(w')$ hence $\#_1(w) = \#_1(0w'1) = 1 + \#_1(w') \geqslant 1 + \#_0(w') = \#_0(w)$;
- $S \Rightarrow S1 \Rightarrow^n w'1$: By IH $\#_1(w') \geqslant \#_0(w')$ hence $\#_1(w) = \#_1(w'1) = 1 + \#_1(w') \geqslant \#_0(w') = \#_0(w)$.

Parse Trees

Parse trees are a way to represent derivations.

A parse tree reflects the internal structure of a word of the language.

That is, using parse trees it becomes very clear which is the variable that was replaced at each step.

In addition, it becomes clear which terminal symbols where generated/derived form a particular variable.

Parse trees are very important in compiler theory.

In a compiler, a *parser* takes the source code into its parse tree.

This parse tree is the structure of the program.

Parse Trees

Definition: Let $G = (V, T, \mathcal{R}, S)$ be a CFG. The *parse trees* for G are trees with the following conditions:

- Nodes are labelled with a variable;
- Leaves are either variables, terminals or ϵ;
- If a node is labelled A and it has children labelled $X_1, X_2, \ldots, X_n$ respectively from left to right, then it must exist in $\mathcal{R}$ a production of the form $A \to X_1 X_2 \ldots X_n$.

Note: If a leaf is ϵ it should be the only child of its parent A and there should be a production $A \to \epsilon$.

Note: Of particular importance are the parse trees with root S.

Exercise: Construct the parse trees for $0 == E + 1$ and for 001100.

Concrete and Abstract Syntax

Concrete syntax describes the way documents are written while *abstract syntax* describes the pure structure of a document.

The abstract syntax of some data is its structure described as a data type.

A parse tree also describe the structure of the data but they may contain features such as parentheses which are syntactically significant but that are implicit in the structure of the abstract syntax tree.

Example: Abstract Syntax of Simple Expressions

Given the grammar

$$\begin{array}{lcl} E & \rightarrow & 0 \mid 1 \mid E + E \mid E * E \mid \text{if } B \text{ then } E \text{ else } E \mid (E) \\ B & \rightarrow & \text{True} \mid \text{False} \mid E < E \mid E == E \end{array}$$

its abstract syntax can be defined by the following data types:

```
data Exp = Z | O | Plus Exp Exp | Mult Exp Exp |
           IfThenElse BExp Exp Exp

data BExp = T | F | Less Exp Exp | Eq Exp Exp

bexp = Less Z (Plus O (Plus Z Z))
```

Height of a Parse Tree

Definition: The *height* of a parse tree is the maximum length of a path from the root of the tree to one of its leaves.

Observe: We count the *edges* in the tree, and not the number of nodes and the leaf in the path.

A path consisting simply of a leaf is 0.

Example: The height of the parse tree for $0 == E + 1$ is 3 and the one for 001100 is 5.

Yield of a Parse Tree

Definition: A *yield* of a parse tree is the string resulted from concatenating all the leaves of the tree from left to right.

Observations:

- We will show later than the yield of a tree is a string derived from the root of the tree;
- Of particular importance are the yields that consist of only terminals; that is, the leaves are either terminals or ϵ;
- When, in addition, the root is S then we have a parse tree for a string in the language of the grammar;
- We will see that yields can be used to describe the language of a grammar.

Inference, Derivations and Parse Trees

Given a CFG $G = (V, T, \mathcal{R}, S)$ we will show the following equivalences:

1. The recursive inference procedure determines that string w is in the language of the variable A;
2. $A \Rightarrow^* w$;
3. $A \overset{lm}{\Rightarrow}^* w$;
4. $A \overset{rm}{\Rightarrow}^* w$;
5. There is a parse tree with root A and yield w.

Note: The equivalences of 2–5 are also valid when the string w contains variables.

Note: Showing $3 \Rightarrow 2$ and $4 \Rightarrow 2$ is trivial.

To show the equivalences we will prove $1 \Rightarrow 5$, $5 \Rightarrow 3$, $5 \Rightarrow 4$ and $2 \Rightarrow 1$.

Overview of Next Lecture

Sections 5.2.3–5.2.6 and 5.4:

- Inference, derivations and parse trees;
- Ambiguity in grammars;
- Chomsky hierarchy.