# Finite Automata and Formal Languages

**TMV026/DIT321 – LP4 2010**

**Lecture 10**

**April 27th 2010**

**Overview of today's lecture:**

- **Context-Free Grammars**



## Context-Free Grammars

Not all languages are regular languages as we have seen.

***Context-free grammars*** (CFG) define what we will call ***context-free languages***.

We have that (we will show this)

regular languages $\subseteq$ 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, $\mathcal{L} = \{w \in \Sigma^* \mid w = \mathsf{rev}(w)\}$.

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

Example of palindromes over $\Sigma$ are $\epsilon$, 0110, 00011000, 010.

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

- $\epsilon$, 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 and those representing boolean expressions.

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





## 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

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

**Example:** Simple expressions can be defined as

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





## Context-Free Grammars

**Definition:** A ***context-free grammar*** is a 4-tuple $G = (V, T, R, S)$ 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.)
- $R$ 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 "$\to$"
  - A string of 0 or more terminal 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
- $w, x, y, z, \ldots$ are string of terminals
- $X, Y, \ldots$ are either terminal of variables
- $\alpha, \beta, \gamma, \ldots$ 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. We can do it in 2 different (equivalent) ways:

**Recursive inference:** From body to head.

**Example:** We infer that 0110110 is a palindrome

| | | | |
|---|---|---|---|
| 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:** From head to body. We need to use a new relation $\Rightarrow$.

**Example:** We infer that $0 == 0 + 1$ is a boolean expression

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





## Formal Definition of Derivation

Let $G = (V, T, R, S)$ be a CFG.

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

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:** We saw in the slide before that $B \Rightarrow^* 0 == 0 + 1$.





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

**Example:**

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

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

**Example:**

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





## Sentential Forms

Derivations from the start variable have an special role.

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

(We talk of *left sentential form* if $S \overset{lm}{\Rightarrow}{}^* \alpha$ and *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
$B \overset{rm}{\Rightarrow} E == E \overset{rm}{\Rightarrow} E == E + E \overset{rm}{\Rightarrow} E == E + 1 \overset{rm}{\Rightarrow} E == 0 + 1$

## 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 $\gamma$ 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 $A \Rightarrow X_1 X_2 \ldots X_k \Rightarrow^* w$, then we can break $w$ into pieces $w_1, w_2, \ldots, w_k$ 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 $X_1 X_2 \ldots X_k \Rightarrow^* \alpha$ then all positions of $\alpha$ 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
$E \Rightarrow E+E \Rightarrow 0+E \Rightarrow 0+1$.





## Language of a Grammar

**Definition:** Let $G = (V, T, R, S)$ 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***





## 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 Step:** 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. The $\mathcal{L}(G)$ is the set of palindromes over $\{0,1\}$.*

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

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

- If $|w| = 0$ or $|w| = 1$ then $w$ is $\epsilon$, 0 or 1.

We have productions $P \to \epsilon$, $P \to 0$ and $P \to 1$. Then $P \Rightarrow^* w$ so $w \in \mathcal{L}(G)$.

- Assume $|w| > 1$. Since $w = \mathsf{rev}(w)$ then $w$ should start and finish with the same symbol. Then $w = 0w'0$ or $w = 1w'1$. Also, $w'$ should be a palindrome.

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 on the number of steps in the derivation that $w$ is a palindrome.

- 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.

- 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 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 (that is, $w' = \mathsf{rev}(w')$) and then so is $w$.





## 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 $G = (V, T, R, S)$ 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 $\epsilon$
- 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 a production of the form $A \to X_1 X_2 \ldots X_n$.

**Note:** If a leaf is $\epsilon$ 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 $S$.

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





## 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 on a leaf is 0.

**Example:** The height of the parse tree for $0 == E + 1$ is 3 and the one for 001100 is 4.





## 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 $\epsilon$.
- 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 $G = (V, T, R, S)$ 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 $2 \Rightarrow 1$.

## From Recursive Inference to Parse Trees ($1 \Rightarrow 5$)

**Theorem:** *Let $G = (V, T, R, S)$ be a CFG. If the recursive inference tells us that $w$ is the language of $A$ then there is a parse tree with root $A$ and yield $w$.*

Proof: By induction on the number of steps in the recursive inference.

- 1 step: Then we have a production $A \to w$ and it is trivial to build the tree.

- $n+1$ steps: Our IH is that the statement is true for strings $x$ and variables $B$ such that $x$ is in the language of $B$ can be inferred in at most $n$ steps.

Suppose the last step in the inference uses the production $A \to X_1 X_2 \dots X_k$.

We break $w$ up as $w_1 w_2 \dots w_k$ where ...





## From Recursive Inference to Parse Trees ($1 \Rightarrow 5$) (Cont.)

We break $w$ up as $w_1 w_2 \dots w_k$ where:

1. If $X_i$ is a terminal then $w_i = X_i$.
2. If $X_i$ is a variable then $w_i$ is the string which was inferred to be in the language of $X_i$. This inference took at most $n$ steps and we can apply IH. Then we have a tree with root $X_i$ and yield $w_i$.

We can now construct a a tree with root $A$, first line of children $X_1, X_2, \dots, X_k$, and the trees given by the IH when $X_i$ is a variable or the corresponding terminal when $X_i$ is a terminal.

The yield of the tree is the concatenation of the yields of the children of $A$ which is $w_1 w_2 \dots w_k = w$.





## From Trees to Leftmost Derivations ($5 \Rightarrow 3$)

**Theorem:** *Let $G = (V, T, R, S)$ be a CFG. If there is a parse tree with root $A$ and yield $w$ then $A \overset{lm}{\Rightarrow}{}^* w$ in $G$.*

Proof: By induction in the height of the tree.

- Height 1: There must be a production $A \to w$ and then $A \overset{lm}{\Rightarrow} w$ and $A \overset{lm}{\Rightarrow}{}^* w$.

- Height $n > 1$: Let $X_1, X_2, \dots, X_k$ be the children of the root $A$. Note that there must be a production $A \to X_1 X_2 \dots X_k$. Now:

1. If $X_i$ is a terminal, define $w_i = X_i$.
2. If $X_i$ is a variable, then it is the root of a subtree. Let $w_i$ be the yield of $X_i$. This subtree has height less than $n$ and then by IH $X_i \overset{lm}{\Rightarrow}{}^* w_i$.

Observe that $w = w_1 w_2 \dots w_k$.





## From Trees to Leftmost Derivations ($5 \Rightarrow 3$) (Cont.)

To construct a leftmost derivation for $w$ we start at $A \overset{lm}{\Rightarrow} X_1 X_2 \dots X_k$.

By induction on $i$ we show that $A \overset{lm}{\Rightarrow}{}^* w_1 w_2 \dots w_{i-1} X_i X_{i+1} \dots X_k$.

- For $i = 0$ we already have $A \overset{lm}{\Rightarrow} X_1 X_2 \dots X_k$.

- Assume $A \overset{lm}{\Rightarrow}{}^* w_1 w_2 \dots w_{i-1} X_i X_{i+1} \dots X_k$.

1. If $X_i$ is a terminal we do nothing. So $A \overset{lm}{\Rightarrow}{}^* w_1 w_2 \dots w_{i-1} w_i X_{i+1} \dots X_k$.
2. If $X_i$ is a variable, by IH we have that $X_i \overset{lm}{\Rightarrow} \alpha_1 \overset{lm}{\Rightarrow} \alpha_2 \dots \overset{lm}{\Rightarrow} w_i$.
   Apply this sequence of derivations to go from
   $A \overset{lm}{\Rightarrow}{}^* w_1 w_2 \dots w_{i-1} X_i X_{i+1} \dots X_k$ to $A \overset{lm}{\Rightarrow}{}^* w_1 w_2 \dots w_{i-1} w_i X_{i+1} \dots X_k$.
   Observe that each step is a leftmost derivation!

When $i = k$ then we have constructed a leftmost derivation
$A \overset{lm}{\Rightarrow}{}^* w_1 w_2 \dots w_{i-1} w_i w_{i+1} \dots w_k$.

## From Trees to Rightmost Derivations ($5 \Rightarrow 4$)

**Theorem:** *Let $G = (V, T, R, S)$ be a CFG. If there is a parse tree with root $A$ and yield $w$ then $A \overset{rm}{\Rightarrow}^* w$ in $G$.*

Proof: The proof is similar to that of the previous theorem.

The difference is that we first need to expand $X_k$ then $X_{k-1}$ and so on until we get to $X_1$.





## From Derivations to Recursive Inference ($2 \Rightarrow 1$)

**Theorem:** *Let $G = (V, T, R, S)$ be a CFG. If $A \Rightarrow^* w$ then the recursive inference procedure applied to $G$ determines that $w$ is in the language of $A$.*

Proof: By induction on the length of the derivation of $A \Rightarrow^* w$.

- 1 step: Then $A \to w$ is a production and the basis part of the recursive inference will find that $w$ is in the language of $A$.

- $n+1$ steps: Our IH is that the statement holds for any derivation of at most $n$ steps.

  Let $A \Rightarrow X_1 X_2 \dots X_k \Rightarrow^* w$ be the first step.

  Now, we can break $w$ as $w = w_1 w_2 \dots w_k$ where ...





## From Derivations to Recursive Inference ($2 \Rightarrow 1$) (Cont.)

Now, we can break $w$ as $w = w_1 w_2 \dots w_k$ where:

1. If $X_i$ is a terminal then $X_i = w_i$.
2. If $X_i$ is a variable then $X_i \Rightarrow^* w_i$. This derivation takes at most $n$ steps and then by IH $w_i$ is inferred to be in the language of $X_i$.

If $A \Rightarrow X_1 X_2 \dots X_k \Rightarrow^* w$ then there must be a production $A \to X_1 X_2 \dots X_k$ and we know that each $w_i$ is either $X_i$ or belongs to the language of $X_i$.

The last step of the recursive inference procedure must use these facts to obtain that $w_1 w_2 \dots w_k$ is in the language of $A$.