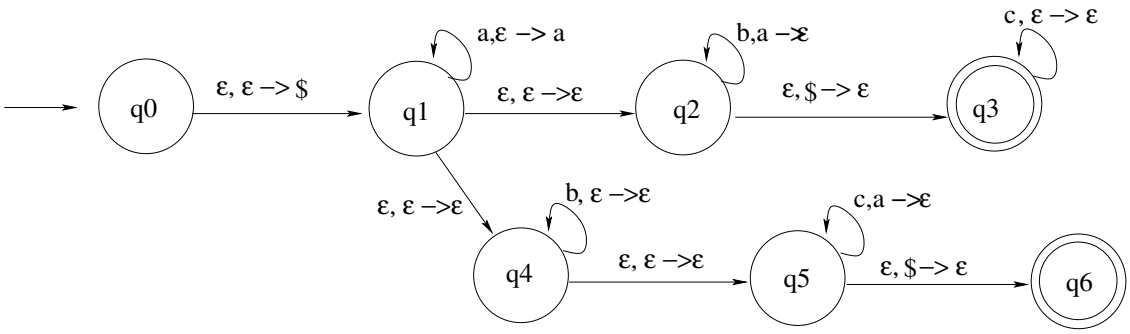
CS 3719 (Theory of Computation and Algorithms) – Lecture 10

Antonina Kolokolova*

January 27, 2011

Let us do one more example of a Pushdown Automaton. In this case, we will consider a language where non-determinism is really unavoidable: $\{a^i b^j c^k | i = j \vee i = k\}$.

Example 1. The following PDA recognizes the language $\{a^i b^j c^k | i = j \vee i = k\}$.



Here, the tape alphabet is $\Gamma = \{a, \$\}$. Note the non-deterministic choice this PDA makes: in the state q_1 it forks between the automaton matching letters b to letters a on the stack, or matching the c's.

1 Context-Free Grammars

We have shown earlier that regular languages can be described by regular expressions. It is natural to ask is there a similar description for the languages computed by pushdown automata. Indeed there is, and it is a natural formalism that has been used for a long time on its own: context-free grammars. Noam Chomsky defined them as one of the types in his hierarchy; the context-free grammar syntax has been used in the programming language community to describe the syntax of programming languages since Algol (known there as Backus-Naur Form).

*The material in this set of notes came from many sources, in particular "Introduction to Theory of Computation" by Sipser and course notes of U. of Toronto CS 364.

When you think of a grammar the first thing that might come to mind is studying a natural language such as French in school, and all these rules about nouns and verbs. Indeed, it is possible to do much of natural language processing (although not everything) using context-free grammar formalism.

Example 2. Consider the following rules:

$< sentence >$	$- > < nounphrase >< verbphrase >$
$< verbphrase >$	$- > < verb > \mid < verb >< nounphrase >$
$< nounphrase >$	$- >$`Jane`$\mid$`the assignment`
$< verb >$	$- >$`solved`$\mid$`did`$\mid$`decided`

For example, a sentence "Jane did the assignment" can be generated by this grammar, and also "Jane solved". On the other hand, this grammar can generate "The assignment solved Jane", which might not be a desired result.

Now that we have seen an example, let's define formally what is a context-free grammar.

Definition 8. *A* context-free grammar *(CFG) is a 4-tuple* (V, Σ, R, S)*, where*

1) *V is a finite set of variables, with $S \in V$ the start variable.*
2) *Σ is a finite set of* `terminals` *(disjoint from the set of variables).*
3) *R is a finite set of rules, with each rule consisting of a variable followed by $- >$ followed by a string of variables and terminals.*

A grammar is a set of substitution rules, where a variable at the head of a rule can be substituted by the string in the body of a rule whenever that variable occurs. Let $A \to w$ be a rule of the grammar, where w is a string of variables and terminals. Then A can be replaced in another rule by w: uAv in a body of another rule can be replaced by uwv (we say uAv yields uwv,denoted $uAv \Rightarrow uwv$). If there is a sequence $u = u_1, u_2, \ldots u_k = v$ such that for all i, $1 \leq i < k$, $u_i \Rightarrow u_{i+1}$ then we say that u derives v (denoted $v \overset{*}{\Rightarrow} v$.)

Definition 9. *If G is a context-free grammar, then the language of G is the set of all strings of terminals that can be generated from the start variable: $\mathcal{L}(G) = \{w \in \Sigma^* | S \overset{*}{\Rightarrow} w\}$. A language is called a* context-free language *(CFL) if there exists a CFG generating it.*

In the above example, variables are $< sentence >, < nounphrase >, < verbphrase >$, and $< verb >$, with the start variable $S =< sentence >$, the $\Sigma =$ { `Jane, the assignment, solved,did,decided` } and R consists of the rules listed above. To be more precise, there

are more rules than 4: we used a shortcut "—" for several different rules starting with the same variable. E..g, we should have had two rules $< nounphrase > - >$ `Jane`, and $< nounphrase > - >$ `the assignment`.

To see how a CFG can generate a nonregular language, consider the language $\{0^n1^n\}$ for which we constructed a PDA in the last lecture.

Example 3. The following CFG generates the language $\{0^n1^n\}$:

$A - > 0A1|\epsilon$

A derivation of a string 000111 in this grammar is:

$A - > 0A1 - > 00A11 - > 000A111 - > 000111$

The last step is because adding or removing an empty substring ϵ does not change a string.

We often visualize a derivation as a tree (a parse tree), with variables as internal nodes and terminals as leaves.

Example 4. Consider the following grammar. Here is a parse tree for a string 0110101. Note that there are several possible parse trees for this string. In this case, we say that a grammar is *ambiguous*.

$A \to AA$
$A \to BAB$
$B \to OB$
$B \to \epsilon$
$A \to 1$