CS 3719 (Theory of Computation and Algorithms) – Lecture 9

Antonina Kolokolova*

January 25, 2011

1 Pushdown automata

Last class we have seen that some languages are not regular. And, moreover, the examples we have seen are languages that can easily be computed by a simple algorithm in any modern programming language. Now a natural question is whether it is possible to add a little extra power to NFAs, so that the resulting model of computation would be able to handle languages such as $\{0^n q^n\}$. Indeed, it is possible to do so by giving NFAs a little "ability to count", some unlimited memory. There are several ways of adding memory to NFAs, and we will look at two of them, Pushdown Automata and Turing machine (in short, Pushdown Automata have an access to an unlimited stack, and Turing machines to an unlimited tape.) In this lecture, we will look at Pushdown automata and analyze its power. Later we will show that Pushdown Automata still fall short of computing many languages that we view as easily computable by our usual algorithmic techniques.

Informally, a pushdown automaton is just an NFA with a stack. So the additional part of the description of such an automaton should include transitions that operate with the stack, as well as stack alphabet.

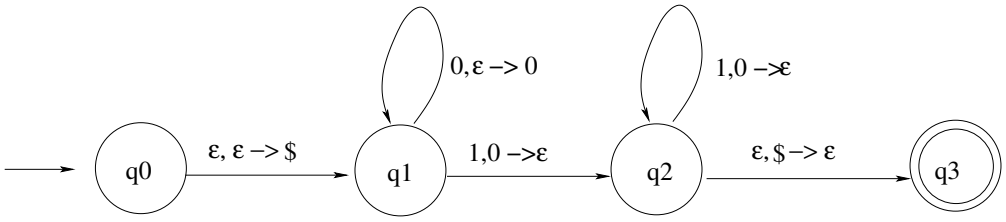
Definition 7. *A pushdown automaton (PDA) is a 6-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, F)$ *where* Q *is the set of states,* Σ *the input alphabet,* Γ *the stack alphabet,* q_0 *the start state,* F *is the set of finite states and the transition function* $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\epsilon\}))$.

That is, each transition of a PDA pops a symbol (possibly ϵ, corresponding to popping nothing) off the stack; δ specifies which set of states to go from the current state on reading an input symbol and a stack symbol, and for every such state, which, if any, symbol to push onto the stack.

*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.

Note that this definition extend the definition of a non-deterministic finite automaton. It is possible to define deterministic pushdown automata by similarly extending the definition of a DFA. However, there are languages accepted by non-deterministic PDAs that no deterministic one can accept (such as a set of palindromes). The proof of this is beyond the scope of this course; however, it is good to remember this as a case where non-determinism actually results in a more powerful model of computation: it didn't for the finite automata. Later in the course we will talk about the P vs. NP problem, a major open problem in computer science which asks about the role of non-determinism for feasible computation.

Example 1. Recall the language $\{0^n 1^n\}$ which we have shown to be non-regular in previous lecture. The following PDA accepts this language.



Here, the tape alphabet is $\Gamma = \{0, \$\}$, where \$ is a special symbol used as an empty stack marker. Since we never need to put 1 on the stack in this PDA, it is OK not to have 1 in Γ, although it is common to take $\Sigma \subseteq \Gamma$.