DFA & PDA

Alex S.[*]

1 Regular Languages

Many languages are infinite, and we need a way to describe them in finite ways. One way is to start with a simple element of a language, and show how we can construct other elements of that language using simple string operations. The set of languages we will examine now are called regular languages, and are generated by three basic operations: catenation, union, and Kleene* (the Kleene star).

The Kleene* signifies repetition of something that preceded it 0 or more times. For example, $a*$ can generate λ (if it repeats 0 times), a if it repeats 1 time, aa if it repeats twice, and so on and so on.

1.1 Regular Expressions

Regular expressions are basically patterns that allow us to describe regular languages. The operators are: + for union, and * for Kleene star. Catenation is just two symbols next to each other.

For example, we can have: $a(a+b)*ba$, which basically says "a, followed by (a or b) repeated any number of times, then followed by b followed by a."

1.2 Finite Automata

A finite automaton, or FA for short (or DFA—for Deterministic Finite Automaton) is a conceptual "machine" for recognizing regular languages.

FA is a 5 tuple - $(Q, \sum, q_0, A, \delta)$, where Q is the set of all states, $\sum$ is the alphabet, q_0 is the starting state, A is the accepting states, δ is the transition function.

A compiler generally specifies its lexical analyzer using regular expressions, which are translated into a finite automaton (a program/library to turn stream of characters into tokens).

[*]alex@theparticle.com

1.3 Nondeterminism

Finite Automatons don't have to be deterministic—the transition, the δ can send you to any number of states. These are called NFAs (for Non-deterministic Finite Automaton). There is also NFA-λ, that can make a state transition without reading any characters at all.

The reason for NFA and NFA-λ is to more easily work with finite automatons. It is usually much easier to specify a non-deterministic finite automaton than a deterministic one. It is however possible to turn any non-deterministic FA into a deterministic one—after all, they both describe *regular languages*.

λ-closure is a method of eliminating the λ transitions, while sub-set construction is a method of constructing a DFA out of an NFA.

1.4 Kleene's Theorem

Kleene's Theorem basically says that since both regular languages and finite automatons describe regular languages, we can convert one to another using a well defined automated procedure (which is the theorem). Examples in class.

2 Context Free Languages

Pal (language of palindromes) is context free.

Same as regular languages, only with recursion. Have no context.

2.1 Context Free Grammars

Grammar is a 4-tuple, $(V, \sum, S, P)$, where V is a set of variables, $\sum$ is a set of terminals, S is the starting variable (S$\in V$), and P is a set of formulas of the form A-¿ something, where A is element of V, and something is element of (V union $\sum$)*

derivation trees, ambiguity

left recursion, common prefixes

getting rid of lambdas

2.2 Push-Down Automata

PDA is a 7-tuple, $(Q, \sum, \Gamma, q_0, Z_0, A, \delta)$, where Q are states, $\sum$ is input alphabet, Γ is the stack alphabet, q0 is the first state, Z0 is the initial stack symbol (element of Γ), A is a set of accepting states (subset of Q), and $\delta : Q \times (\sum \cup \{\lambda\}) \times \Gamma$ (finite subsets of $Q \times \Gamma *$.

We move from "configuration" to configuration.

configuration of PDA is current state, input string, and top of stack.