Illinois Institute of Technology
Department of Computer Science

Lecture 24: April 22, 2019

CS 330 Discrete Structures
Spring Semester, 2019

1 Regular Expressions

Now we define a *regular expression*, which turns out to be equivalent to finite automata, but can be a nicer way to express some regular languages (those that are accepted by regular expressions and/or finite automata). The empty set ($\emptyset$) is a regular expression, corresponding to the language containing no words. Epsilon (ϵ) is a regular expression corresponding to the language containing only the empty word. (Notice that there is a (somewhat subtle) difference between these two.) Also, each letter x in our alphabet is a regular expression corresponding to the language containing that letter as its only word. We also define regular expressions to be closed under union, concatenation, and Kleene star. Using these properties together, it is possible to define quite complicated languages.

Since we have shown that regular languages have the same closure properties, it is perhaps not surprising that our "new" class of languages turns out to be exactly equivalent to the old one:

Theorem: Every regular expression represents a regular language, and every regular language can be represented by a regular expression.

Proof: By the rules of construction of regular expressions, the first part of this theorem is clear.

To prove the second part of the theorem, we use a variant on the Floyd-Warshall algorithm[1] to construct a regular expression given an arbitrary finite state machine.

Note that this algorithm assumes that the machine has exactly one final state. If it has multiple final states, the algorithm can be run once for each and the results joined with the + (union) operator on regular expressions.

Say the machine has n states. Give each state a unique integer between 1 and n, where state 1 is the initial state and state n is the final state. Define the expression R_{ij}^k to be the regular expression describing strings that carry the FSM from state i to state j, using only states between 1 and k as intermediate states. Then the expression that corresponds to the machine as a whole is R_{1n}^n.

R_{ij}^k is defined recursively:

$$\begin{aligned} R_{ij}^0 &= \{x \in \Sigma | \delta(i,x) = j\} \\ R_{ij}^k &= R_{ij}^{k-1} + R_{ij}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1} \end{aligned}$$

As a regular expression can be constructed given any FSM, any regular language can be expressed by a regular expression.

[1]The Floyd-Warshall algorithm, presented in section 25.2 of *Introduction to Algorithms*, 3nd edition, by Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein, MIT Press, 2009., is a $\Theta(|V|^3)$ dynamic-programming algorithm for solving the all-pairs shortest-paths problem on a directed graph. Here we apply it to an FSM, treated as a graph.