Lecture 6: Regular Expressions

Michael Engling

Department of Computer Science
Stevens Institute of Technology

CS 334 Automata and Computation
Fall 2015

OUTLINE: Regular Expressions

Regular Expressions

Given an alphabet Σ we can construct Regular Expressions inductively:

1. If $a \in \Sigma$, then a is a regular expression,
2. ε is a Regular Expression,
3. $\emptyset$ is a Regular Expresson.

These three form our "Base Cases" for the inductive definition.
(Think of them as "virtual Legos".)

Now, given Regular Expressions R_1 and R_2:

4. $R_1 \cup R_2$ is a Regular Expression,
5. $R_1 \circ R_2$ is a Regular Expression, and
6. R_1^* is a Regular Expression.

Order of Precedence

Just as $3 + 4 \times 5 = 23$ rather than 60 because we give $\times$ higher precedence than $+$, we rank our three Regular Operations: The Kleene Star takes precedence over concatenation which takes precedence over union.

Hence $a \cup b \circ a^*$ differs from $((a \cup b) \circ a)^*$.

Furthermore, we will frequently have need to express non-empty, arbitrary strings. We express possibly empty arbitrary strings with Σ^*. We can guarantee at least one symbol with $\Sigma\Sigma^* = \Sigma^+$, for convenience.

Examples

Given $\Sigma = \{0, 1\}$:

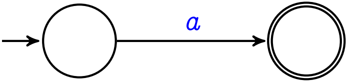
1. $0^*10^* = \{w | w \text{ contains a single } 1\}$.
2. $\Sigma^*1\Sigma^* = \{w | w \text{ contains at least one } 1\}$.
3. $\Sigma^*001\Sigma^* = \{w | w \text{ contains substring } 001\}$.
4. $1^*(01^+)^* = \{w | \text{ every } 0 \text{ in } w \text{ is followed by a } 1\}$.
5. $(\Sigma\Sigma)^* = \{w | w \text{ is a string of even length.}\}$
6. $(\Sigma\Sigma\Sigma)^* = \{w | \text{ the length of } w \text{ is a multiple of } 3\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w | w \text{ start and end in same symbol }\}$.
9. $(0 \cup \varepsilon)1^* = \{01^*, 1^*\}$.
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{01, 0, 1, \varepsilon\}$. (FOIL)
11. $1^*\emptyset = \emptyset$. The empty set annihilates anything concatenated with it.
12. $\emptyset^* = \{\varepsilon\}$. Take no strings as many times as you like, and you get an empty string.

Regular Expressions and Regular Languages

A language is regular iff a regular expression describes it.

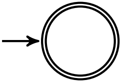
It is easy to see that a language described by a regular expression is regular:

If $a \in \Sigma$, then:



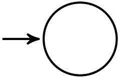
accepts a.

2. Furthermore:



accepts ε, and

3. Given the empty set:



Regular Expressions and Regular Languages

And we have seen over the last week(s) that, given regular languages R_1 and R_2...

4. We can construct an NFA for $R_1 \cup R_2$ (closure property), and
5. We can construct an NFA for $R_1 \circ R_2$ (closure property), and
6. We can construct an NFA for R_1^* (closure property).

Hence, given a regular expression, we can construct an NFA that recognizes the language it represents. Hence, the language it represents is regular.

Generalized Nondeterministic Finite Automaton

In order to do the other direction of our proof, we must show that any DFA corresponds to a regular expression.

We will accomplish this by introducing the notion of a Generalized NFA.

Given a DFA, convert it to an NFA as we did last time. Next, augment it with two new states, a new start state with ε-edge to the old start state and a single new final state with ε-edges entering it from (only) the old final states.

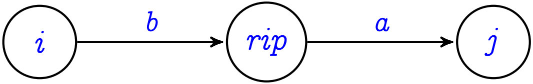
If multiple edges from state i (labeled a) to state j (labeled b), replace them with a single edge with a union labeled $a \cup b$.

Finally, add edges between states that had no edges, and label them $\emptyset$.

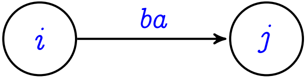
(We will omit this step in the examples that follow.)

Three Basic Cases: Case 1

We will systematically reduce our $(n+2)$-state GNFA to a 2-state GNFA wherein the only remaining label is the regular expression recognized by the GNFA.

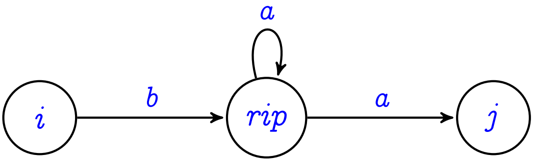


As we remove state rip, we leave ba as an edge:

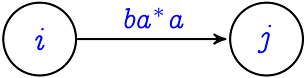


Three Basic Cases: Case 2

We will systematically reduce our $(n+2)$-state GNFA to a 2-state GNFA wherein the only remaining label is the regular expression recognized by the GNFA.

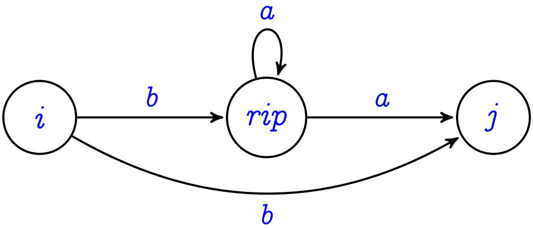


As we remove state rip, we leave ba^*a as an edge:

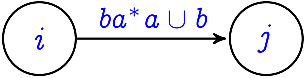


Three Basic Cases: Case 3

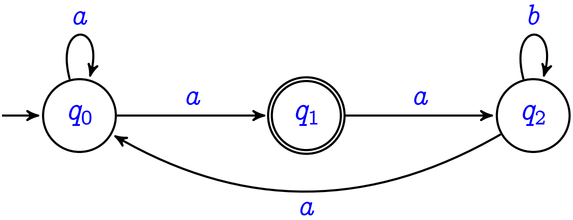
We will systematically reduce our $(n+2)$-state GNFA to a 2-state GNFA wherein the only remaining label is the regular expression recognized by the GNFA.



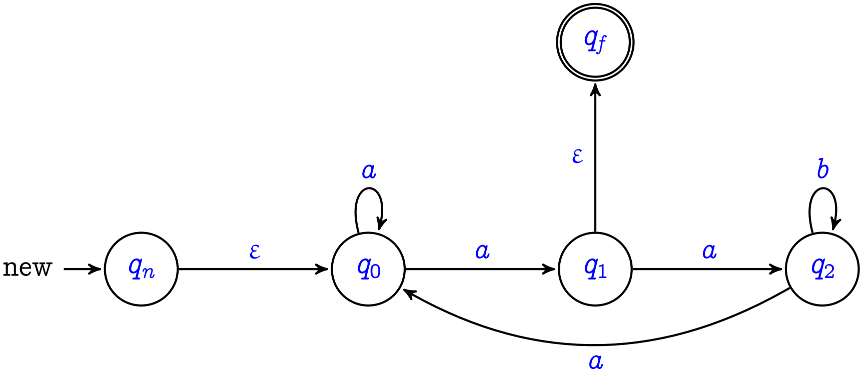
As we remove state rip, we leave $ba^*a \cup b$ as an edge:



Dare We Attempt An Example...?



Dare We Attempt An Example...?



Two correct Regular Expressions:

$a^+(ab^*a^+a)^*$ (Removal orders: q_0, q_2, q_1 and q_2, q_0, q_1)

$(a^2b^*a \cup a)^*a$ (Removal order: q_1, q_2, q_0)