CMSC 141 Automata and Language Theory

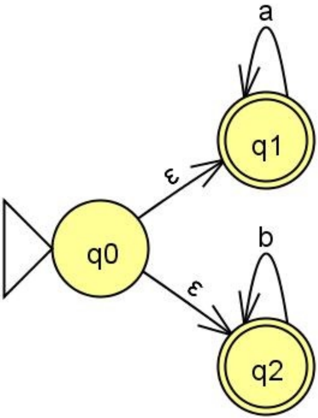
Regular Languages

Mark Froilan B. Tandoc

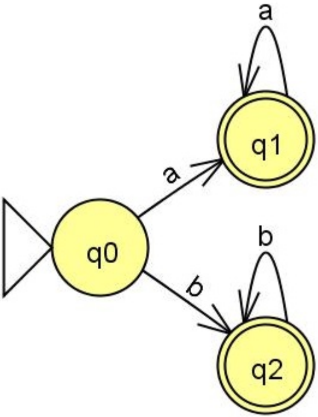
August 29, 2014

Another example of εNFA

$L = \{w | w \text{ is all a's or all b's}\}$



What does this automata accept?

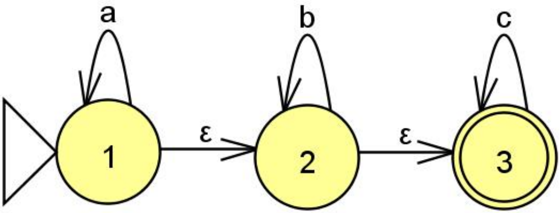


ε-closure

- Informally, the ε-closure of a state q is the set of all states reachable from q via ε-moves
- A recursive definition of the ε-closure:
  - $q \in \varepsilon$-close(q)
  - if $p \in \varepsilon$-close(q) and $r \in \delta(p, \varepsilon)$
  - then $r \in \varepsilon$-close(q)

Example of ε-closures

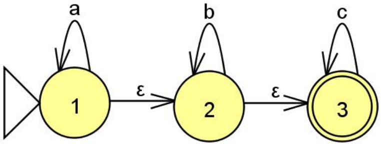
$$\{w | w \text{ is sorted}\} \text{ over } \Sigma = \{a, b, c\}$$



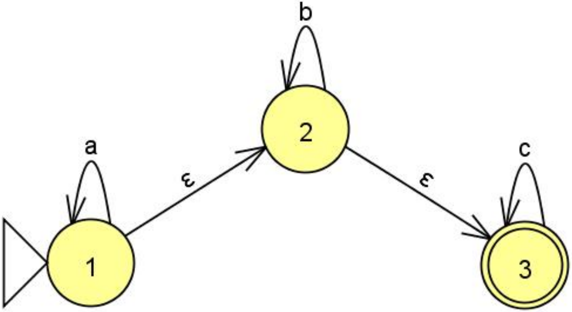
- ε-close$(1) = \{1,2,3\}$
- ε-close$(2) = \{2,3\}$
- ε-close$(3) = \{3\}$

Elimination of ε-moves

We can use the closure of the states to remove the ε-moves

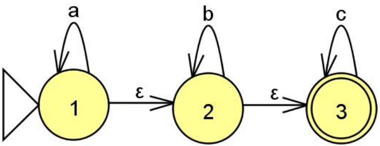


- ε-close$(1) = \{1,2,3\}$
- ε-close$(2) = \{2,3\}$
- ε-close$(3) = \{3\}$

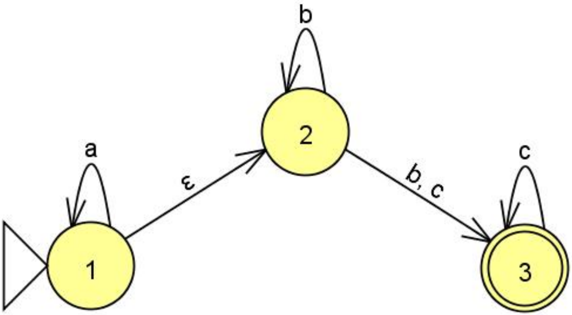


Elimination of ε-moves

We can use the closure of the states to remove the ε-moves

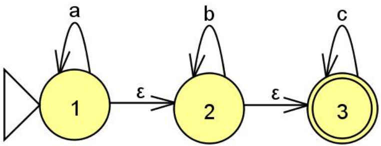


- ε-close(1) $= \{1,2,3\}$
- ε-close(2) $= \{2,3\}$
- ε-close(3) $= \{3\}$

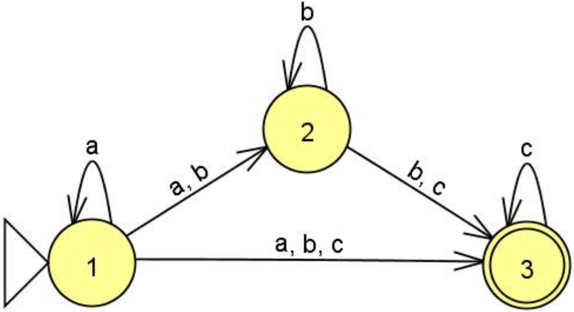


Elimination of ε-moves

We can use the closure of the states to remove the ε-moves

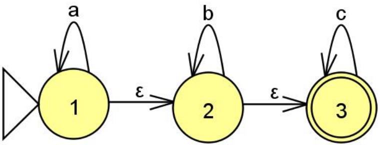


- ε-close$(1) = \{1,2,3\}$
- ε-close$(2) = \{2,3\}$
- ε-close$(3) = \{3\}$

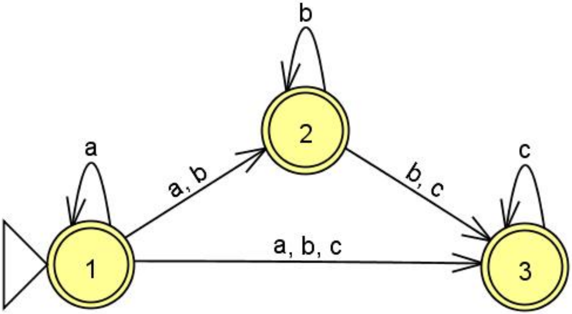


Elimination of ε-moves

$F' = F \cup \varepsilon\text{-close}(q_0)$, if $\varepsilon\text{-close}(q_0)$ contains some final state



- $\varepsilon\text{-close}(1) = \{1,2,3\}$
- $\varepsilon\text{-close}(2) = \{2,3\}$
- $\varepsilon\text{-close}(3) = \{3\}$



Regular Expression (regex)

- Sequence of characters or symbols that can be treated as string patterns.
- Can be used to describe languages. The set of strings that matches the regex pattern will be the language.
- e.g. $(a + b)^*$

RegEx Operations

Given an alphabet Σ, we can define some constants as regular expressions:

- $\emptyset$ denoting $\emptyset$
- ε denoting $\{\varepsilon\}$
- literal character from Σ, say a, denoting $\{a\}$

RegEx Operations

Concatenation

Let R and S be regular expressions, RS denotes the set of strings obtained by concatenating all strings in R to all strings in S.

Examples:

- a $\Rightarrow$ {a}, b $\Rightarrow$ {b}
  ab $\Rightarrow$ {ab}
- ab $\Rightarrow$ {ab}, c $\Rightarrow$ {c}
  abc $\Rightarrow$ {abc}

RegEx Operations

Alternation

Let R and S be regular expressions, $R + S$ denotes the union of R and S.

Examples:

- $\text{a} \Rightarrow \{\text{a}\}$, $\text{b} \Rightarrow \{\text{b}\}$
  $\text{a} + \text{b} \Rightarrow \{\text{a}, \text{b}\}$
- $\text{ab} \Rightarrow \{\text{ab}\}$, $\text{c} \Rightarrow \{\text{c}\}$
  $\text{ab} + \text{c} \Rightarrow \{\text{ab}, \text{c}\}$

RegEx Operations

Kleene Star

Let R be a regular expression, R^* denotes the set of all strings that can be made by concatenating the strings described by R with themselves any number of times (including zero).

Examples:

- a $\Rightarrow$ {a}
  a* $\Rightarrow$ {ε, a, aa, aaa, ...}
- ab $\Rightarrow$ {ab}
  (ab)* $\Rightarrow$ {ε, ab, abab, ababab, ...}

RegEx Operations

RegEx also observe hierarchy of operations.

- Parenthesis
- Kleene star
- Concatenation
- Alternation

More Complex Examples

- $(a + b)^* =$
  $\{\varepsilon$, a, b, aa, ab, ba, bb, aaa, aab, ...$\}$
- $a^* + b^* =$
  $\{\varepsilon$, a, b, aa, bb, aaa, bbb, aaaa, ...$\}$
- $a^*b^* =$
  $\{\varepsilon$, a, b, ab, aaa, aab, abb, bbb, aaaa, ...$\}$
- $ab^*a =$
  $\{$aa, aba, abba, abbba, abbbba, ...$\}$
- $a(a+b)a^* =$
  $\{$aa, ab, aaa, aba, aaaa, abaa, aaaaa, ...$\}$
- $(a+ab)^* =$
  $\{\varepsilon$, a, ab, aa, aab, abab, aaa, aaab, aaba, ...$\}$

Quiz

Give the regular expression of following language:
$L = \{\varepsilon, ab, ba, abab, baba, ababab, bababa, ...\}$
Answer: (ab)* $+$ (ba)*

References

- Previous slides on CMSC 141
- M. Sipser. Introduction to the Theory of Computation. Thomson, 2007.
- J.E. Hopcroft, R. Motwani and J.D. Ullman. Introduction to Automata Theory, Languages and Computation. 2nd ed, Addison-Wesley, 2001.
- E.A. Albacea. Automata, Formal Languages and Computations, UPLB Foundation, Inc. 2005
- JFLAP, www.jflap.org