Theory of Formal Languages and Automata

Lecture 6

Mahdi Dolati

Sharif University of Technology

Fall 2024

February 17, 2024

Regular Expressions

- Regular operations: Build up expressions
- Value of a regular expression is a language
  - $(\{0\} \cup \{1\}) \circ \{0\}^* = (0 \cup 1)0^*$.
  - Set of all strings start with a zero or one followed by any number of zeros.
- Powerful method for describing patterns.
- Priority: Star, concatenation, union.

Example

- $(0 \cup 1)^*$
- Let $\Sigma = \{0, 1\}$:
  - Σ: Length-one strings
  - Σ^*: All possible strings
  - $0\Sigma^*$: All strings start with a zero
  - Σ^*1: All strings ends with a one

Regular Expressions

Definition (Regular Expression)

R is a regular expression if:

1. $a \in \Sigma \rightarrow$ Language$=\{a\}$
2. $\varepsilon \rightarrow$ Language$=\{\varepsilon\}$. This language contains one string, the empty string.
3. $\emptyset \rightarrow$ Language$=\{\}$. This language does not contain any strings.
4. R_1 and R_2 are regular expressions:
   1. $(R_1 \cup R_2)$
   2. $(R_1 \circ R_2)$
   3. (R_1^*)

- Circular definition vs. inductive definition

Regular Expressions

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$
- $R \cup \varepsilon \neq R, \qquad \exists R$

$$R = 0 \rightarrow L(R) = \{0\} \tag{1}$$
$$R = 0 \rightarrow L(R \cup \varepsilon) = \{0, \varepsilon\} \tag{2}$$

- $R \circ \emptyset \neq R, \qquad \exists R$

$$R = 0 \rightarrow L(R) = \{0\} \tag{3}$$
$$R = 0 \rightarrow L(R \circ \emptyset) = \emptyset \tag{4}$$

Regular Expressions

Example

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

- $0^*10^* = \{w | w \text{ contains a single } 1\}$
- $\Sigma^*1\Sigma^* = \{w | w \text{ has at least one } 1\}$
- $\Sigma^*001\Sigma^* = \{w | 001 \text{ is a substring of } w\}$
- $1^*(01^+)^* = \{w | \text{ every } 0 \text{ is followed by a } 1 \text{ in } w\}$
- $(\Sigma\Sigma)^* = \{w | \text{ length of } w \text{ is even}\}$
- $(\Sigma\Sigma\Sigma)^* = \{w | \text{ length of } w \text{ is a multiple of } 3\}$
- $01 \cup 10 = \{01, 10\}$

Example

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

- $0\Sigma^*0 \cup 1\Sigma^*10 \cup 1 = \{w|w$ starts and ends with the same symbol$\}$
- $(0 \cup \varepsilon)1^* = 01 \cup 1^*$
- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
- $1^*\emptyset = \emptyset$
- $\emptyset^* = \{\varepsilon\}$

Example

- Regular expressions: A useful tool in the *design of compilers*
- Tokenization: Extract tokens
- Generate the lexical analyzer
- Example:
  - $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$$(+ \cup - \cup \varepsilon)(D^{+} \cup D^{+}.D^{*} \cup D^{*}.D^{+}) \tag{5}$$

  - Remember operator priorities

Regular Expression and DFA Equivalence

Theorem

A language is regular iff a regular expression can describe it.

Lemma

If a regular expression describe a language, then it the language is regular.

Proof idea:

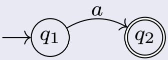
- Assume we have a regular expression R describing language A. We convert R into an NFA that recognizes A.
- Remember: DFA and NFA are equivalent.

Regular Expression and DFA Equivalence

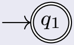
Proof.

Consider six cases in the definition of regular expressions:

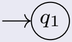
1. $a \in \Sigma \rightarrow$ Language=$\{a\}$.
   - $N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$
   - $\delta(q_1, a) = \{q_2\}$
2. $\varepsilon \rightarrow$ Language=$\{\varepsilon\}$.
   - $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$
   - $\delta(q_1, b) = \emptyset$
3. $\emptyset \rightarrow$ Language=$\{\}$.
   - $N = (\{q_1\}, \Sigma, \delta, q_1, \emptyset)$
   - $\delta(q_1, b) = \emptyset$
4. R_1 and R_2 are regular expressions (use closure proofs):
   - 4.1. $(R_1 \cup R_2)$
   - 4.2. $(R_1 \circ R_2)$
   - 4.3. (R_1^*)



(a) Case 1



(b) Case 2



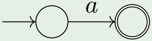
(c) Case 3

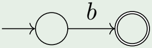
□

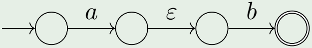
Regular Expression and DFA Equivalence

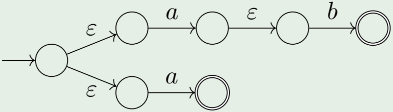
Example

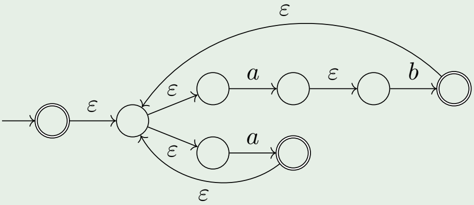
$(ab \cup a)^*$

$a :$ 

$b :$ 

$ab :$ 

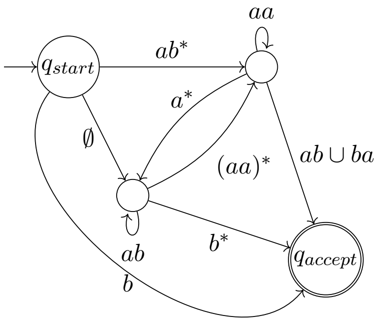
$ab \cup a :$ 

$(ab \cup a)^* :$ 

Lemma

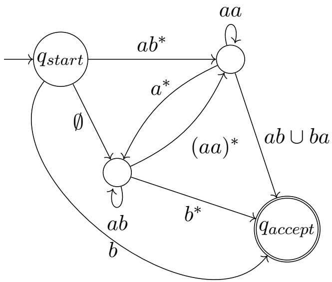
If a language is regular, then a regular expression can describe it.

- Generalized Nondeterministic Finite Automaton
  - DFS $\rightarrow$ GNFA $\rightarrow$ regular expression
  - GNFA is a NFA that its arrows labels may be regular expressions
  - GNFA reads blocks of symbols



Regular Expression and DFA Equivalence

- Special form of GNFAs:
  - One start and one accept state, different from each other
  - One arrow from start to all other states and no incoming arrows
  - One arrow to accept from all other states and no outgoing arrows
  - One arrow between each pair of states
  - One self-loop in each state



Regular Expression and DFA Equivalence

Definition (GNFA)

A GNFA is a 5-tuple, $(Q, \Sigma, \delta, q_{start}, q_{accept})$, where

1. Finite set of states: Q,
2. Alphabet: Σ,
3. Transition function: $\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow \mathcal{R}$,
   - $\mathcal{R}$ is the set of all regular expressions over Σ.
4. Start state: q_{start}
5. Accept state: q_{accept}

Regular Expression and DFA Equivalence

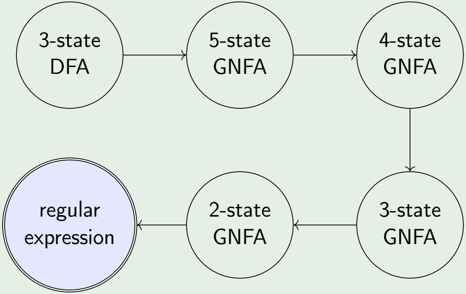
- GNFA accepts $w = w_1 w_2 \ldots w_k$, if there exists a sequence of states $q_0, q_1, \ldots, q_k$ such that:
  1. $q_0 = q_{start}$
  2. $q_k = q_{accept}$
  3. for each i, we have $w_i \in L(\delta(q_{i-1}, q_i))$.

Regular Expression and DFA Equivalence

- Convert DFA to GNFA:
  - A new start to connect to old start with ε arrow
  - A new accept to be connected from all old accepts with with ε arrows
  - Combine arrows with union
  - Use $\emptyset$ for arrows not in DFA
- Convert GNFA to regular expression:
  - Assume the GNFA has k states
  - Convert the GNFA to an equivalent GNFA with $k-1$ states
  - If $k=2$ there is one arrow from a start state to an accept state
  - Label of the remaining arrow is the regular expression

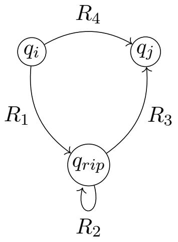
Regular Expression and DFA Equivalence

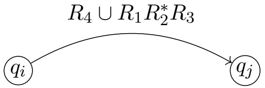
Example



Regular Expression and DFA Equivalence

- How to convert a GNFA to an equivalent GNFA with one less state:
  - Select a state other than the start and accept at random
  - Call the selected state q_{rip}
  - Remove q_{rip}
  - Update the label of remaining arrows to compensate for the absence of q_{rip}
  - New labels: Describe all string that change the state the machine either or via q_{rip}





- `CONVERT`(G)
  1. $k \leftarrow$ The number of states of G
  2. If $k = 2$, then G has one start state, one accept state, and a single arrow labeled R. Return R.
  3. If $k > 2$, then select $q_{rip} \in Q - \{q_{start}, q_{accept}\}$ and construct $G^{'} = (Q^{'}, \Sigma, \delta^{'}, q_{start}, q_{accept})$, where:
     1. $G^{'} = Q - \{q_{rip}\}$ and
     2. for $q_i \in Q^{'} - \{q_{accept}\}$ and $q_j \in Q^{'} - \{q_{start}\}$:

$$\delta^{'}(q_i, q_j) = (R_1)(R_2)^{*}(R_3) \cup (R_4), \tag{6}$$

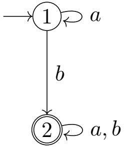
where,

$$R_1 = \delta(q_i, q_{rip}), \qquad R_2 = \delta(q_{rip}, q_{rip}), \tag{7}$$

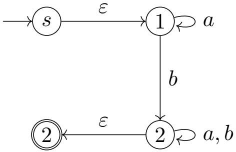
$$R_3 = \delta(q_{rip}, q_j), \qquad R_4 = \delta(q_i, q_j). \tag{8}$$

  4. Return `CONVERT`($G^{'}$)

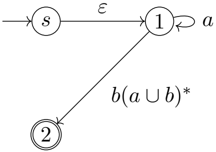
Regular Expression and DFA Equivalence



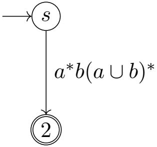
(a) Input DFA



(b) Constructed GNFA



(c) After one `CONVERT`



(d) After two `CONVERT`s