FORMAL LANGUAGES AND AUTOMATA, 2025 FALL SEMESTER

Lec 04. Regular expression

Eunjung Kim

Regular expression

We want to compactly describe the 'pattern' of the following languages using union, concatenation and Kleene star operations.

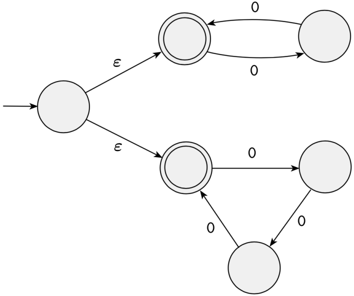


Figure 1.33, Sipser 2012.

1. The set of all $0, 1$ strings with exactly one 1's.
2. The set of all $0, 1$ strings with at least one 1's.
3. The set of strings over Σ of even length.

Formal definition of Regular expression

Regular expression over a finite alphabet Σ

Regular expression over Σ is a string consisting of symbols of Σ, parenthesis ( ), and the operators $\cup, \circ, ^*$ that can be generated as follows.

- Each symbol $x \in \Sigma \cup \{\epsilon\}$ is a regular expression.
- $\emptyset$ is a regular expression.
- $(R_1 \cup R_2)$ is a regular expression if R_1 and R_2 are regular expressions.
- $(R_1 \circ R_2)$ is a regular expression if R_1 and R_2 are regular expressions.
- R^* is a regular expression if R is a regular expression

Examples of Regular expression

Assume $\Sigma = \{0, 1\}$. Which language does the regular expression describe?

1. 0^*10^*.
2. $\Sigma^*1\Sigma^*$.
3. $1^*(01^+)^*$.
4. $(\Sigma\Sigma)^*$.
5. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$.
6. $1^*\emptyset = \emptyset$.
7. $\emptyset^* = \{\epsilon\}$.

Regular language

Value of a regular expression

For a regular expression R, the set of all strings which <u>can be generated following the expression</u> is denoted by $\mathcal{L}(R)$, said to be <u>the language of R</u>.

$\mathcal{L}(R)$ is also called the value of R, or the language described by R.

For two regular expressions R_1 and R_2, the value of union / concatenation / kleene star is...

- $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$.
- $L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$.
- $L(R_1^\star) = L(R_1)^\star$.

Equivalence of regular expression and finite automata

Regular expression=NFA

A language $A \subseteq \Sigma^*$ is described by a regular expression if and only it is a regular language, i.e. recognized by some NFA.

One direction can be proved easily using what we learnt in the previous lecture. Which direction is it?

Equivalence proof: easy direction

Equivalence Theorem, Part I

If a language $A \subseteq \Sigma^*$ is described by a regular expression, then there is a (nondeterminist) finite automata M such that $L(M) = A$.

Proof idea: inductively build an NFA from NFAs accepting each symbol, $\{\epsilon\}$ or $\emptyset$ by applying each regular operations (union, concatenation, Kleene star).

Equivalence proof: easy direction

Constructing NFA recognizing the language $(\{\epsilon\} \cup \{a\} \cup \{ab\})^\star$.

Constructing NFA recognizing the language $(0 \cup 1)^\star 10$.

Equiv proof: the other direction

Equivalence Theorem, Part II

If a language $A \subseteq \Sigma^*$ is recognized by a finite automata A, then there is a regular expression R such that $\mathcal{L}(R) = A$.

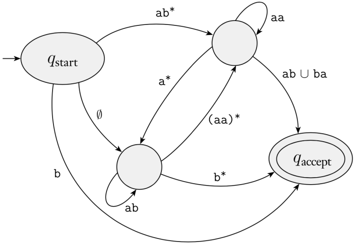
Proof idea: we shall see a procedure which converts DFA (or NFA) recognizing A into a regular expression.

Equiv proof: the other direction

Generalized Nondeterministic Finite Automata

- Generalized NFA is a NFA in which each arc carries a regular expression as a label.
- There are a unique source q_{start} as an initial state and a unique sink q_{accept} as an accept state.
- Between any other states there are arcs in both ways, including loops.

Figure 1.61, Sipser 2012.



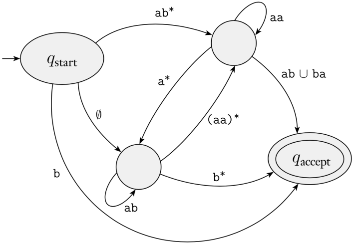
Equiv proof: the other direction

A formal description of GNFA

- Generalized NFA is a 5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$, where
- the transition function maps $(Q \setminus q_{accept}) \times (Q \setminus q_{start})$ to the set of all regular expressions over Σ.

In the transition diagram, we often omit an arc which carries $\emptyset$.

Figure 1.61, Sipser 2012.



EQUIV PROOF: THE OTHER DIRECTION

A GNFA ACCEPTS A STRING w

If w can be written as $w_1 w_2 \cdots w_\ell$, $w_i \in \Sigma^*$, and there is a sequence of states $r_0, \ldots, r_\ell$ such that

- r_0 is the initial state of GNFA,
- w_i is in the value of $\delta(r_{i-1}, r_i)$, i.e. $w_i \in L(\delta(r_{i-1}, r_i))$, and
- r_ℓ is the accept state of GNFA.

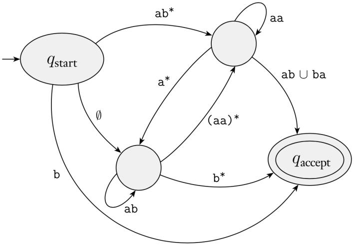
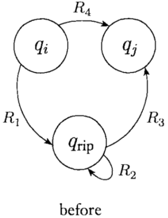


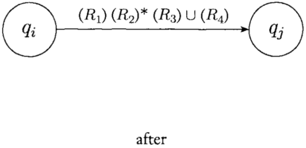
Figure 1.61, Sipser 2012.

EQUIV PROOF: THE OTHER DIRECTION

Proof idea.

- Initial NFA can be seen as GNFA, possibly after a simple modification to ensure a unique accept state, no incoming/outgoing arc from the unique initial/accept state.
- Reduce the number of states of GNFA inductively by eliminating a state of $Q - \{q_{start}, q_{accet}\}$ one by one, each elimination yielding an equivalent GNFA.
- The final GNFA with two states q_{start} and q_{accept} carries a single regular expression, a desired end product.
- How to eliminate a state q_k and update the label on (q_i, q_j):





Example: from NFA to regular expression

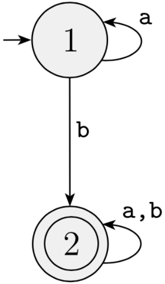


Figure 1.67 (a), Sipser 2012.

Equiv proof: the other direction

Proof by induction. Let G' be a GNFA obtained by eliminating state q_k from GNFA G. It suffices to prove that G accepts a string w if and only if G' does.

- Let $r_0, \ldots, r_\ell$ be a sequence of states appearing in the accepting computation history for $w = w_1 \cdots w_\ell$ in G.
- If q_k does not appear in this sequence, done.

Equiv proof: the other direction

Proof by induction. Let G' be a GNFA obtained by eliminating state q_k from GNFA G. It suffices to prove that G accepts a string w if and only if G' does.

- Let $r_0, \ldots, r_\ell$ be a sequence of states appearing in the accepting computation history for $w = w_1 \cdots w_\ell$ in G.
- If q_k does not appear in this sequence, done.
- If not, consider a maximal subsequence $r_a, \ldots, r_b$ of contiguous occurrences of q_k. Note that $1 \leq a \leq b < \ell$.

Equiv proof: the other direction

Proof by induction. Let G' be a GNFA obtained by eliminating state q_k from GNFA G. It suffices to prove that G accepts a string w if and only if G' does.

- Let $r_0, \ldots, r_\ell$ be a sequence of states appearing in the accepting computation history for $w = w_1 \cdots w_\ell$ in G.
- If q_k does not appear in this sequence, done.
- If not, consider a maximal subsequence $r_a, \ldots, r_b$ of contiguous occurrences of q_k. Note that $1 \leq a \leq b < \ell$.
- Observe: $w_a \in L(\delta(r_{a-1}, r_a))$, $w_{i+1} \in L(\delta(r_i, r_{i+1})) = L(\delta(q_k, q_k))$ for every $i \in [a, b-1]$, and $w_{b+1} \in L(\delta(r_b, r_{b+1}))$.

Equiv proof: the other direction

Proof by induction. Let G' be a GNFA obtained by eliminating state q_k from GNFA G. It suffices to prove that G accepts a string w if and only if G' does.

- Let $r_0, \ldots, r_\ell$ be a sequence of states appearing in the accepting computation history for $w = w_1 \cdots w_\ell$ in G.
- If q_k does not appear in this sequence, done.
- If not, consider a maximal subsequence $r_a, \ldots, r_b$ of contiguous occurrences of q_k. Note that $1 \leq a \leq b < \ell$.
- Observe: $w_a \in L(\delta(r_{a-1}, r_a))$, $w_{i+1} \in L(\delta(r_i, r_{i+1})) = L(\delta(q_k, q_k))$ for every $i \in [a, b-1]$, and $w_{b+1} \in L(\delta(r_b, r_{b+1}))$.
- That is: $w_a \cdots w_{b+1}$ is described by

  $$\delta(r_{a-1}, r_a) \cdot \delta(q_k, q_k)^\star \cdot \delta(q_k, r_{b+1})$$

  which is in the union expression $\delta'(r_{a-1}, r_{b+1})$ in the G'.

Equiv proof: the other direction

Conversely,

- Let $r_0, \ldots, r_\ell$ be a sequence of states appearing in the accepting computation history for $w = w_1 \cdots w_\ell$ in G'.
- Each w_i in is the language of $\delta'(r_{i-1}, r_i)$.
- $\delta'(r_{i-1}, r_i) = \delta(r_{i-1}, r_i) \cup \delta(r_{i-1}, q_k) \cdot \delta(q_k, q_k)^\star \cdot \delta(q_k, r_i)$.
- Therefore, $w_i \in \delta(r_{i-1}, r_i)$ or
  w_i can be written as $x_1 \cdots, x_m$ such that
  1. $x_1 \in \delta(r_{i-1}, q_k)$
  2. $x_2 \in \delta(q_k, r_i)$

  or

  1. $x_1 \in \delta(r_{i-1}, q_k)$
  2. $x_j \in \delta(q_k, q_k)$ for each $2 \leq j < m$
  3. $x_m \in \delta(q_k, r_i)$
- Now replace the computation history in G' by...... to obtain a computation history in G.

Example: from NFA to regular expression

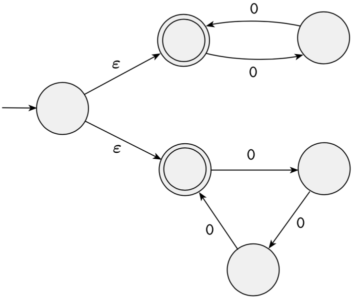


Figure 1.33, Sipser 2012.