Finite Automata and Regular Expressions

SITE: http://www.info.univ-tours.fr/~mirian/
Theorem

If $L = L(A)$ for some DFA, then there is a regular expression $R$ such that $L = L(R)$.

- We are going to construct regular expressions from a DFA by eliminating states.
- When we eliminate a state $s$, all the paths that went through $s$ no longer exist in the automaton.
- If the language of the automaton is not to change, we must include, on an arc that goes directly from $q$ to $p$, the labels of paths that went from some state $q$ to state $p$, through $s$.
- The label of this arc can now involve strings, rather than single symbols (may be an infinite number of strings).
- We use a regular expression to represent all such strings.
- Thus, we consider automata that have regular expressions as labels.
Constructing a regular expression from a finite automaton

1. For each accepting state \( q \), apply the reduction process to produce an equivalent automaton with regular expression labels on the arcs. Eliminate all states except \( q \) and the start state \( q_0 \).

2. If \( q \neq q_0 \), then we shall be left with a two-state automata:

   ![Automaton Diagram](image)

   One regular expression that describes the accepted strings: \((R + SU^*T)^* SU^*\)

3. If the start state is also a final state, then we are left with a one-state automaton and the regular expression denoting strings that it accepts is \( R^* \)

4. The desired regular expression is the union of all the expressions derived from the reduced automata for each accepting states.
Example

Example: NFA accepts strings of 0 and 1 such that either the second or the third position from the end has a 1. Represented by the regular expression

$$(0 + 1)^* 1 (0 + 1) + (0 + 1)^* 1 (0 + 1) (0 + 1)$$
Theorem

Every language defined by a regular expression is also defined by a finite automaton.

- Suppose $L = L(R)$ for a regular expression $R$. We show that $L = L(E)$ for some $\varepsilon-NFA$ $E$ with
  1. Exactly one accepting state
  2. No arcs into the initial state
  3. No arcs out of the accepting state

- The proof is by structural induction on $R$, following the recursive definition of regular expressions.
Proof

Basis

The basis of the construction of fsa from regular expressions:

1. Expression $\epsilon$: the language of the FSA is $\{\epsilon\}$.
2. Expression $\emptyset$: $\emptyset$ is the language of FSA.
3. Expression $a$: the language of the FSA is $\{a\}$.

All these automata satisfies the three initial conditions
Proof

Induction
The inductive step of the construction of fsa from regular expressions

1. The expression is $R + S$ for some smaller expressions $R$ and $S$.
2. The expression is $RS$ for some smaller expressions $R$ and $S$.
3. The expression is $R^*$ for some smaller expression $R$.
4. The expression is $(R)$ for some smaller expression $R$. 