Equivalence of Regular Expressions and FSMs

Greg Plaxton
Theory in Programming Practice, Spring 2005
Department of Computer Science
University of Texas at Austin
Regular Language

- Recall that a language is a (possibly infinite) set of strings over some specified alphabet.

- For any FSM $M$, the *language accepted by $M$* is the set of all strings accepted by $M$.

- A language is *regular* if it is equal to the set of strings defined by some regular expression.

- We’ll prove that regular expressions and FSMs are equivalent in power:
  - Every FSM accepts a regular language.
  - Every regular language is accepted by some FSM.
Every FSM Accepts a Regular Language

• Fix an FSM \( M \), and number the states of \( M \) from 1 to \( n \) arbitrarily

• Let \( R_{ij}^k \) denote the set of strings that cause \( M \) to transition from state \( i \) to state \( j \) without visiting an intermediate state numbered higher than \( k \)

• Claim: For all \( i \), \( j \), and \( k \), \( R_{ij}^k \) is a regular language
  – We’ll prove the claim by induction on \( k \geq 0 \)
  – Why does the claim imply the desired result?

• The key observation for carrying out the induction step is that

\[
R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}
\]
Every Regular Language is Accepted by Some FSM

- We prove this result in three stages:
  - First, we define the notion of a nondeterministic FSM and prove that deterministic and nondeterministic FSMs are equivalent in power
  - Second, we define the notion of a nondeterministic FSM with \( \epsilon \)-transitions and prove that nondeterministic FSMs with and without \( \epsilon \)-transitions are equivalent in power
  - Finally, we prove that every regular language is accepted by some nondeterministic FSM with \( \epsilon \)-transitions
Nondeterministic Finite State Machines

- A nondeterministic FSM is the same as a (deterministic) FSM except that the transition function maps each state/symbol pair to a (possibly empty) subset of the states, as opposed to a single state.

- When we run a nondeterministic machine on a given input string, we repeatedly choose the next state arbitrarily from the subset specified by the transition function.
  - An execution is *good* if the transition function never specifies an empty subset from which to choose the next state; otherwise, it is *bad*.
  - A nondeterministic FSM $M$ is said to accept a string $x$ if there exists some good execution of $M$ on string $x$ that terminates in an accepting state.
Simulation of a Nondeterministic FSM by a (Deterministic) FSM

• Fix a nondeterministic FSM $M$ and let $S$ denote the set of states of $M$

• We simulate $M$ via an FSM $M'$ with $2^{|S|}$ states, one corresponding to each subset of $S$

• The key idea is to define the transition function of $M'$ so that the following condition holds for any input $x$
  – The execution of $M'$ on input $x$ terminates in the state of $M'$ corresponding to the set of all states $\alpha$ of $M$ such that some good execution of $M$ on input $x$ terminates in state $\alpha$

• How do we define the accepting states of $M'$ in order to complete the construction properly?
Nondeterministic FSMS with $\epsilon$-Transitions

- A nondeterministic FSM with $\epsilon$-transitions is the same as a nondeterministic FSM except that we allow transitions labeled $\epsilon$ (and called $\epsilon$-transitions) that do not consume an input symbol.

- The notion of a good execution is the same as we had for a nondeterministic FSM without $\epsilon$-transitions, except that whenever we find ourselves in a state with one or more outgoing $\epsilon$-transitions, we have the option to make such a transition without consuming an input symbol.

- Acceptance is then defined in the same way as for nondeterministic FSMs:
  - A nondeterministic FSM $M$ with $\epsilon$-transitions is said to accept a string $x$ if there exists some good execution of $M$ on string $x$ that terminates in an accepting state.
Simulation of a Nondeterministic FSM with $\epsilon$-Transitions by a (Ordinary) Nondeterministic FSM

• Fix a nondeterministic FSM $M$ with $\epsilon$-transitions

• We’d like to create a nondeterministic FSM $M'$ without $\epsilon$-transitions that simulates $M$

• Let the states of $M'$ be the same as the states of $M$

• We define the transition function of $M'$ so that there is a transition from state $\alpha$ to state $\beta$ on input symbol $a$ if and only if $M$ admits a path from $\alpha$ to $\beta$ where one transition is labeled with $a$ and the remaining transitions are $\epsilon$-transitions

• How do we define the accepting states of $M'$ in order to complete the construction properly?
Every Regular Language is Accepted by some Nondeterministic FSM with $\epsilon$-Transitions

- Recall that regular languages are defined in terms of regular expressions.
- We prove the claim by structural induction on the definition of a regular expression.
- Base case: We need to exhibit machines that accept the languages $\emptyset$, \{\epsilon\}, and \{a\} where $a$ is an arbitrary symbol in the alphabet.
- For the induction step, we need to consider regular expressions formed by concatenation, union, and Kleene closure of smaller regular expressions.