COMS W3261
Computer Science Theory
Lecture 6
Decision Problems for Regular Expressions; Mimizing States

Alexander Roth

2014 – 09 – 22

Overview

- Many common decision problems for representations of regular languages are decidable.
- Every regular set has a minimum-state DFA (unique up to renaming of states).

1 Decision Problems for Regular Languages

- We can ask whether a representation of a language has a given property. Such a question is often called a *decision problem.*
- If there is an algorithm to answer the question, we say that problem is *decidable.* For decidable problems we are interested in how quickly a question can be answered as a function of the size of the representation of the language.
- The *emptiness problem* is to decide whether the language denoted by a given representation is empty.
  - Given a finite automaton for a regular language, we can answer the emptiness problem by determining whether there is a path from the start state to a final state. This can be answered in $O(n^2)$ time where n is the number of states in the automaton.
- The *membership problem* is to decide whether a particular string is in the language denoted by a given representation.

- Given a DFA D for a regular language and an input string w, we can answer the membership problem by simulating D processing w beginning in the start state. This can be answered in O($|w|$) time.

1.1 Class Notes

Given a DFA D, is $L(D) = \emptyset$? Is $L(D) = \Sigma^*$?

2 Testing Equivalence of States

- Given a DFA D for a regular language, we say two distinct states p and q
- This says the two states $\delta^*(p, w)$ and $\delta^*(q, w)$ are either both accepting or both nonaccepting.
- If two states of a DFA are not equivalent, then we say they are *distinguishable.*
- Here is what is known as *the table-filing algorithm* for computing all pairs of distinguishable states of a DFA:
  - Input: A DFA $D = (Q, \Sigma, \delta, q_0, F)$.
  - Output: a table T of all pairs of distinguishable states.
  - Method:
    for all states p and q do
      if p is final and q is nonfinal
        add $\{p, q\}$ to T
    for all states p and q do
      for all input symbols a do
        if $\delta(p, a)$ and $\delta(q, a)$ are in T then
          add $\{p, q\}$ to T
    until no more pairs can be added to T
- Theorem: If two states p and q are not distinguishable by the table-filling algorithm, then p and q are equivalent.

2.1 Class Notes

3 Testing Equivalence of DFA's

- We can use the table-filing algorithm to test the equivalence of two DFA's by testing the equivalence of their start states.
- The DFA's are equivalent iff their start states are equivalent.

4 Minimizing the Number of States in a DFA

- We can use the table-filing algorithm as a subroutine to minimize the number of states in a DFA.
- The minimization algorithm:
  - Input: a DFA $A = (Q_A, \Sigma, \delta_A, q_A, F_A)$.
  - Output: an equivalent minimum-state DFA $B = (Q_B, \Sigma, \delta_B, q_B, F_B)$
  - Method:
    1. Eliminate any state that cannot be reached from the start state.
    2. Compute the sets of all equivalent states.
    3. Partition the states into blocks so that
       * all states in the same block are equivalent and
       * no pair of states from different blocks are equivalent.
    4. Construct the minimum-state DFA B as follows:
       (a) Q_B is the set of blocks of equivalent states.
       (b) If R and S are blocks containing the states p and q of A, respectively, then $\delta_B(R, a) = S$ if $\delta_A(p, a) = q$.
       (c) q_B is the block containing q_A.
       (d) A state S is in F_B if S contains a state in F_A.
- Thereom: $L(B) = L(A)$ and no DFA equivalent to A has fewer states than B.

4.1 Class Notes

Input: DFA D

Output: an equivalent DFA with the smallest possible number of states.

Method:
1. Remove all inaccessible states from D.
2. Compute all equivalent states.
3. Partition states into maximal equivalent blocks.
4. From the equivalent blocks, construct a DFA.

What is the fastest way to determine if two regular expressions are equivalent?