CSE211 – Formal Languages and Automata Theory

U1L20 – Decision properties of RL and Introduction Minimization

Dr. P. Saravanan
School of Computing
SASTRA Deemed University

Agenda

- Recap of previous class-Closure properties of RL
- Decision properties of RL
  - Emptiness
  - Membership
- Minimization of DFA

Decision Properties of RL's

- Converting among Representations
  - Assume #symbols = constant and #states = n.
  - From an RE to an automaton (ε-NFA) --- requiring linear time in the size of the RE
  - Conversion from an ε-NFA to a DFA --- requiring $O(n^3 2^n)$ time in the worse cases
  - Conversion from a DFA to an NFA --- requiring $O(n)$ time
  - From an automaton (DFA) to an RE --- requiring $O(n^3 4^n)$ time

Decision Properties of RL's

- Testing Emptiness of RL's
  - Testing if a regular language generated by an automaton is empty:
    - Equivalent to testing if there exists no path from the start state to an accepting state.
    - Requiring $O(n^2)$ time in the worse case.
    - Why? Time proportional to #arcs

    $\Rightarrow$ each state has at most n arcs (to the n states)

    $\Rightarrow$ at most n^2 arcs

    $\Rightarrow$ at most $O(n^2)$ time

Decision Properties of RL's

- Testing Emptiness of RL's
  - A 2-step method for testing if a language generated by an RE is empty:
    - Convert the RE to an ε-NFA ---

      requiring O(s) time as said previously, where

      s = |RE| (length of RE).
    - Test if the language of the ε-NFA is empty ---

      requiring O(n^2) time as said above.
  - The overall time requirement is

    O(s) + O(n^2)

Decision Properties of RL's

- Testing Membership in an RL
  - **Membership Problem**:

    given an RL *L* and a string *w*, is $w \in L$?
  - If *L* is represented by a DFA, the algorithm to answer the problem requires O(*n*) time, where $n = |w|$ (# symbols in the string instead of #states of the automaton).
  - Why? Just processing input symbols one by one to see if an accepting state is reached.

Equivalence & Minimization of Automata

- What we want to show in this section:
  - Testing whether two descriptions of RL's define the same languages.
  - Minimization of DFA's ---
    - Good for implementations of DFA's with less resources (like space, time, IC areas, …)

Equivalence & Minimization of Automata

- Testing Equivalence of States
  - **Goal:**

want to understand when two distinct states *p* and *q* can be replaced by a single state that behaves like both *p* and *q*.

Equivalence & Minimization of Automata

- Testing Equivalence of States
  - Two states are said equivalent if

    for all strings *w*, δ(*p*, *w*) is an accepting state if and only if δ(*q*, *w*) is an accepting state.

  - Note:
    - It is ***not*** necessary to enter the same accepting state for the above definition to be met.
    - We only require that either **both states are accepting** or **both states are non-accepting**.

Equivalence & Minimization of Automata

- Testing Equivalence of States
  - Non-equivalent states are said to be distinguishable.

    That is, state *p* is said to be distinguishable from *q* if there is at least a string *w* such that one of $\hat{\delta}(p, w)$ and $\hat{\delta}(q, w)$ is accepting, and the other is not accepting.
  - A systematic way to find distinguishable states --- use a *table-filling algorithm*

Myhill Nerode Theorem

- Testing Equivalence of States
  - **Table-filling algorithm(**Myhill Nerode Theorem)

Basis.

If p is an accepting state and q is not, then the pair $\{p, q\}$ is distinguishable.

Induction.

- Let p and q be states such that for some input symbol a, the next states $r = \delta(p, a)$ and $s = \delta(q, a)$ are known to be distinguishable. Then $\{p, q\}$ are distinguishable.

Myhill Nerode Algorithm

- Step1: Draw a table for all pairs of states (p,q)
- Step 2: Mark all pairs where p∈F and q∉F
- Step 3: If there are any unmarked pairs (p,q) such that [$\delta(p, a)$, $\delta(q, a)$] is marked, then mark [p,q] where a is an input symbol.
- Repeat this until no more markings can be done
- Step 4: Combine all the unmarked pairs and make them a single state in the minimized DFA

Summary

- Closure properties of RL
  - Closed *Union, Concatenation, Closure (star), Intersection, Complementation, Difference, Reversal, Homomorphism, Inverse homomorphism*
- Decision properties of RL
  - Emptiness
  - Membership
- Minimization of DFA

References

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory*, Languages, and Computation, Pearson, 3rd Edition, 2011.
- Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.

Next Class:

Minimization of DFA: Problems

THANK YOU.