SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 OF THE UGC ACT, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

CSE211 – Formal Languages and Automata Theory

U1L19 – Closure Properties of RL

Dr. P. Saravanan
School of Computing
SASTRA Deemed University

Agenda

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

Closure Properties of RL's

- The term "closure" means "being closed" in the same type of language domain, such as RL's.
- Ex: Addition of integer is closure.
  - Integer+integer=integer
- We will prove a set of "closure" theorems of the form ---
  - "if certain languages are regular, and a language *L* is formed from them by certain operations, then *L* is also regular."

Operations on Languages

- Language operations for the above statement to be true include:
  - *Union*
  - *Concatenation*
  - *Closure (star)*
  - *Intersection*
  - *Complementation*
  - *Difference*
  - *Reversal*
  - *Homomorphism*
  - *Inverse homomorphism*

Closure Properties of RL's

- The union of two regular languages is regular
- The concatenation of regular languages is regular
- The closure (star) of a regular language is regular

Bu definition of Regular Language

- The intersection of two regular languages is regular
- The complement of a regular language is regular
- The difference of two regular languages is regular
- The reversal of a regular language is regular
- A homomorphism(substitution of strings for symbols) of a regular language is regular
- The inverse homomorphism of a regular language is regular

Closure of Regular Languages Under Boolean Operations

The three Boolean operations are union, intersection and complementation

- Let L and M be languages over alphabet Σ. Then L∪M is the language that contains all strings that are in either or both of L and M
- Let L and M be languages over alphabet Σ. Then L∩M is the language that contains all strings that are in both L and M
- Let L be a language over alphabet Σ. Then L the complement of L is the set of strings in Σ^* (universal Language) that are not in L

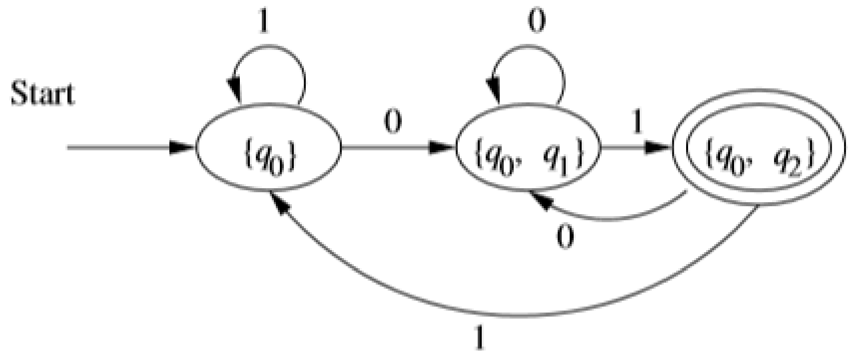
Closed Under Complementation

- Theorem 4.5: If L is a regular language over alphabet then the complement $= \Sigma^* - L$ is also an RL (Σ^* is the universal language)

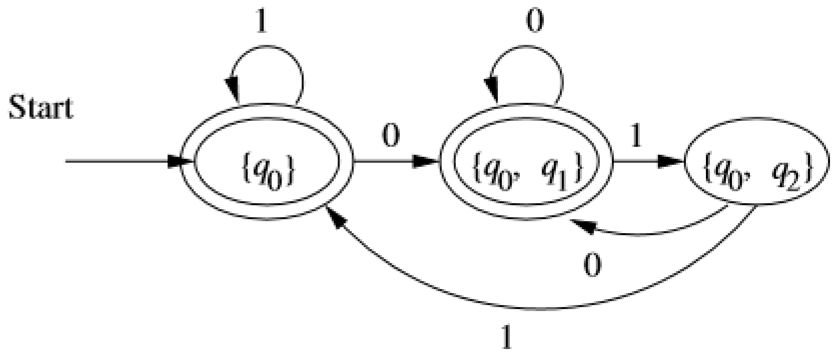
PROOF: Let $L = L(A)$ for some DFA $A = (Q, \Sigma, \delta, q_0, F)$. Then $\overline{L} = L(B)$, where B is the DFA $(Q, \Sigma, \delta, q_0, Q - F)$. That is, B is exactly like A, but the accepting states of A have become nonaccepting states of B, and vice versa. Then w is in $L(B)$ if and only if $\hat{\delta}(q_0, w)$ is in $Q - F$, which occurs if and only if w is not in $L(A)$. $\square$

Closed Under Complementation

- Let A be the automaton of Fig.



- Recall that DFA A accepts all and only the strings of 0s and 1s and ends with 01. In regular expression terms (0+1)*01.
- The Complement is



-

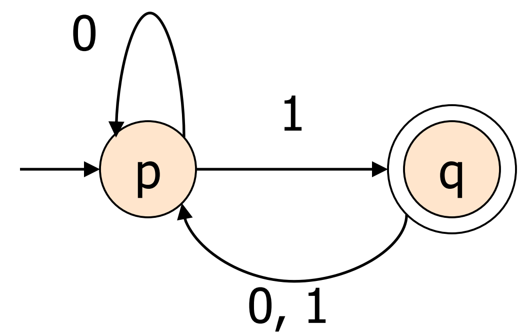
Closed Under Intersection

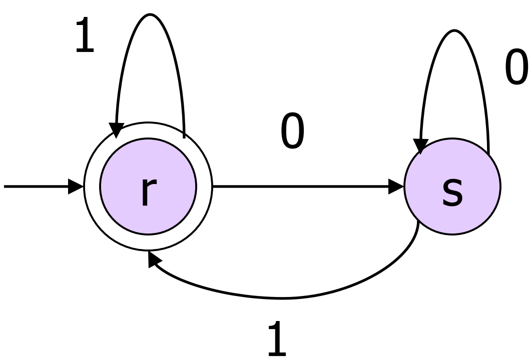
- If L and M are regular languages, then so is L ∩ M.
- Proof:
- Using Demargon's Law
  - L ∩ M = (L 'U M')'
  - = (RE' U RE')'
  - = (RE U RE)'
  - = (RE)'
  - =RE

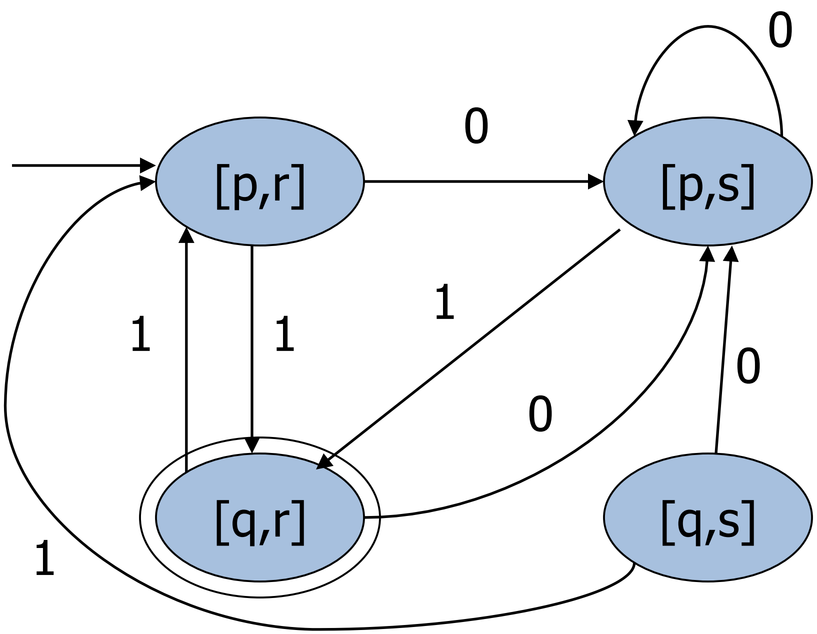
DFA for Intersection (Product Automaton)

- Let A and B be DFA’s whose languages are L and M, respectively.
- Construct C, the product automaton of A and B.
- Make the final states of C be the pairs consisting of final states of both A and B.

Example: Product DFA for Intersection







Closed Under Difference

- Theorem 4.10: If L and M are regular languages then so is L - M = strings in L but not M
- Proof:
  - L – M = $(L \cap M)'$
  - = $(RE \cap RE)'$
  - = $(RE)'$
  - = RE

Closed under Reversal

- The **reversal** of a string $w = a_1a_2, \dots a_n$ is $w^R = a_na_{n-1}\dots a_2a_1$.
- The **reversal** L^R of a language L is the language consisting of the reversals of all its strings.
- **Theorem 4.11:** the reversal L^R of an RL L is also an RL.

Closure under Homomorphism

- A **(string) homomorphism** is a function h which substitutes a particular string for each symbol.

  That is, $h(a) = x$, where a is a symbol and x is a string.

- Given $w = a_1a_2\ldots a_n$, define

$$h(w) = h(a_1)h(a_2)\ldots h(a_n).$$

- Given a language, define

$$h(L) = \{h(w) \mid w \in L\}.$$

- **Theorem 4.14** - If L is an RL, then $h(L)$ is also an RL where h is a homomorphism.

Closure under Homomorphism

- **Example 4.13**
  - Let function h be defined as

$$h(0) = ab \text{ and } h(1) = \varepsilon,$$

then h is a string homomorphism.

  - For examples,

1. $h(0011) = h(0)h(0)h(1)h(1)$

$= abab\varepsilon\varepsilon = abab.$

2. If RE $r = 10^*1$, then $h(L(r)) = L((ab)^*)$.

Closed under inverse homomorphism

- **Inverse homomorphism**:

  Let h be a homomorphism from some alphabet Σ to strings in another alphabet T. Let L be an RL over T. Then $h^{-1}(L)$ is the set of strings w such that $h(w)$ is in L.

- $h^{-1}(L)$ is read "h inverse of L."
- **Theorem 4.16** - If h is a homomorphism from alphabet Σ to alphabet T, and L is an RL, then $h^{-1}(L)$ is also an RL.

Closed under inverse homomorphism

- **Example**
  - Let $L = L((\mathbf{00} + \mathbf{1})^*)$
  - Let (string) homomorphism h be defined as

$$h(a) = 00,\ h(b) = 1.$$

  - It can be proved that

$$h^{-1}(L) = \{\varepsilon,\mathbf{a,b,aa,bb,ab,ba,.....})$$

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.

Summary

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

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

THANK YOU.