COMS3261:
Computer Science Theory

Spring 2013

Mihalis Yannakakis

Lecture 6, 9/23/13

Regular Expressions in UNIX

Shorthands. Geared to Σ = ASCII character set

. (dot) = Σ (. matches any single character)

[a1…ak] = a1 + … ak

[0-9] = any digit ; [a-z] any lower case letter, etc

| used in place of + for union

? = 0 or 1 occurrence: R? = ε+R

+ = 1 or more occurrences: R+ = RR* = R+RR +RRR+…

{n} = n copies: R{3} = RRR

and other features

Algebraic Properties of Operators

- Union (+) is
  - Commutative: $L \cup M = M \cup L$, for any languages L,M
    $R+S \equiv S+R$ for any re's R, S
  - Associative: $(L \cup M) \cup N = L \cup (M \cup N)$
    $(R+S)+T \equiv R+(S+T)$ for any re's R,S,T
  - identity $\varnothing$: $L \cup \varnothing = \varnothing \cup L = L$
    $R + \varnothing \equiv \varnothing + R \equiv R$, for any re R
  - idempotent: $L \cup L = L$
    $R + R \equiv R$
  - mononote (wrt both operands):
    $L \subseteq L' \Rightarrow L \cup M \subseteq L' \cup M$, and $M \subseteq M' \Rightarrow L \cup M \subseteq L \cup M'$

Algebraic Properties of Operators

- Concatenation (.) is
  - Associative: $(L.M).N = L.(M.N)$
    $(R.S).T \equiv R.(S.T)$ for any re's R,S,T
  - identity $\{\varepsilon\}$: $L.\{\varepsilon\} = \{\varepsilon\}.L = L$
    $R.\varepsilon \equiv \varepsilon.R \equiv R$, for any re R
  - annihilator $\varnothing$: $L.\varnothing = \varnothing.L = \varnothing$
    $R.\varnothing \equiv \varnothing.R \equiv \varnothing$, for any re R
  - mononote (wrt both operands):
    $L \subseteq L' \Rightarrow L.M \subseteq L'.M$, and $M \subseteq M' \Rightarrow L.M \subseteq L.M'$

- but not commutative in general: $L.M \neq M.L$

Distributive Property of Operators

- Concatenation distributes over union (+) both from left and right
- L.(M $\cup$ N) = L.M $\cup$ L.N
  R(S+T) $\equiv$ RS+RT for re's
- (M $\cup$ N) . L = M.L $\cup$ N.L
  (S+T)R $\equiv$ SR+TR for re's

Star

- $\emptyset^*= \{\varepsilon\}$; $\{\varepsilon\}^* = \{\varepsilon\}$;
- $L \subseteq L^*$
- Idempotent: $(L^*)^* = L^*$
- Proof:
- $L^* \subseteq (L^*)^*$ because $M \subseteq M^*$ for any language M
- To show other direction $(L^*)^* \subseteq L^*$, let w be any string in $(L^*)^*$

Then $\exists$strings $x_1\ldots x_k$ in L^* such that $w=x_1\ldots x_k \Rightarrow$

$\exists$ strings $y_{11}\ldots y_{1m_1},\ldots, y_{k1}\ldots y_{1mk}$ in L such that $x_1 =y_{11}\ldots y_{1m_1},\ldots$

$x_k = y_{k1}\ldots y_{1m}$ and $w=x_1\ldots x_k \Rightarrow$

$\exists$ strings $y_{11}\ldots y_{1m_1}, y_{k1}\ldots y_{1mk}$ in L such that

$w= y_{11}\ldots y_{1m_1}, y_{k1}\ldots y_{1mk}$

$\Rightarrow w \in L^*$

Algebraic Laws for REs

- If E1, E2 are two regular expressions with variables, the

identity E1≡E2 (often written simply E1=E2) means that if we substitute *any* languages for the variables (or any re's), the resulting languages are equal.

Example: R+S=S+R, R+∅=R etc.

(R*)*=R*

(L+M)*=(L*M*)*

How do we test if an equality is true for all possible languages?

Testing Algebraic Laws

- Take an alphabet that contains a distinct symbol for every variable.
- Substitute the symbol for each variable → concrete re's
- Check if the two re's define the same language
- Theorem: *The test is necessary and sufficient (for expressions that use +,.,* ; if we use other operators, for example complement, it does not hold)*
- Proof:

  - one direction is obvious: if E1=E2 holds for all language substitutions it holds for the particular one

- - other direction: see book

Testing Algebraic Laws

- Theorem: The test is necessary and sufficient (for expressions that use +,.,* ; if we use other operators, for example complement, it does not hold)
- Proof of other direction: see book

key property: given expression E(L1,…,Lm) with variables L1,…,Lm, construct concrete re E(a1,…,am).

If I substitute concrete languages Li in E, then any string w in result can be written as $w=w_1 \ldots w_k$ where each w_i is a string in some L_{ji} and the string $a_{j1} \ldots a_{jk}$ is in E(a1,…,am).

i.e. can obtain all strings of E(L1..Lm) by taking strings of E(a1,…,am) and substituting for each occurrence of each ai some string in Li.

Examples

- LM = ML ?

  Substitute 0 for L , 1 for M → Is 01 =10 ? No!

  This gives a counterexample: L={0}, M={1}
- L(M+N)=LM+LN?

  0(1+2) = 01+02 ? Yes (by definition of concatenation)
- (R+S)* = (R*S*)*

  (0+1)* = (0*1*)*: both are = set of all strings over {0,1}
- R*R* = R*

  0*0* = 0* : both sides are = set of all strings of 0's

Testing Algebraic Laws ctd

- The theorem reduces the testing of algebraic laws (with variables) to the testing of equivalence of concrete regular expressions (without variables)
- We'll see that there is an algorithm for doing this (by transforming to DFA and testing the DFA for equivalence)
- We'll see that testing of DFA equivalence is efficient,

  but the translation to DFA in general blows up the size exponentially.
- For long, complex REs in fact it is not easy to tell if they are equivalent; even telling whether a RE $\equiv \Sigma^*$ is "intractable" (we will discuss such problems at the end of the course).
- But for "small" RE, this is not a problem.

Closure Properties of Regular Languages

Closure Properties

- In general, a set is closed under some operation if the operation applied to any elements in the set yields an element in the set.
- Example: Integers closed under +,-,* but not /
- The class of regular languages closed under Union, Concatenation and * Star
- Proof: use regular expressions: If L, M are any two regular languages, take regular expressions E_L , E_M for them, then $E_L + E_M$ is a regular expression and its language is $L \cup M \Rightarrow L \cup M$ is a regular language
- Language of $E_L . E_M$ is $L.M \Rightarrow L.M$ is regular
- Language of $(E_L)^*$ is $L^* \Rightarrow L^*$ regular

Complement

- Given regular language L over alphabet Σ, its complement L^c (or L-bar) = $\Sigma^* - L$ is also regular.
- Proof: Take DFA $A=(Q,\Sigma,\delta,q_0,F)$ for L.
- Switch accepting and nonaccepting states $\rightarrow$ DFA A'

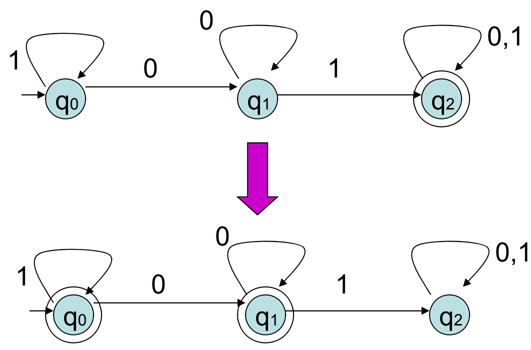
  A word w is accepted by DFA A $\Leftrightarrow \delta\hat{}(q_0,w) \in F \Leftrightarrow$

  $\Leftrightarrow$ word w is not accepted by A'

Example application: All co-finite languages (complements of finite languages) are regular

Example of complementation

L = set of binary strings with substring 01



L = set of binary strings with no substring 01

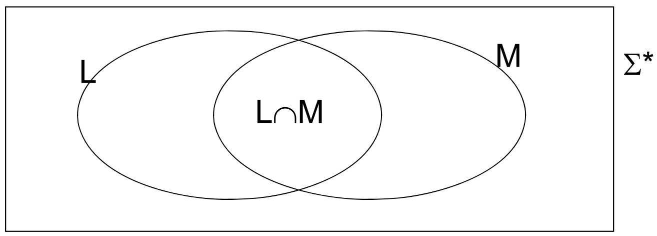
Complement

- Easy to show the closure property and do complementation with DFA, not so easy with NFA or REs.
- Linear complexity for DFA
- Exponential for NFA, RE in general
- i.e. there are examples of a language L with small NFA, RE, but smallest NFA, RE for L^c is exponentially larger

  Example: L = set of strings over {1,…,n} that do not contain all the letters of the alphabet.
- HW: 1. Construct "small" ε-NFA for L (O(n) states) and RE (of size at most $O(n^2)$).
- 2. Argue that every ε-NFA for L must have $\geq 2^n$ states. Conclude that every RE for L must also have size $\geq 2^n$.

Closure under Intersection

- If L, M are regular languages, then $L \cap M$ is also regular
- Proof: $L \cap M = (L^c \cup M^c)^c$
- Regular languages form a Boolean algebra (closed under Boolean operations ($\cup$,$\cap$,compl.)



• Can show closure also by direct construction of a FA for $L \cap M$ from finite automata for L and M

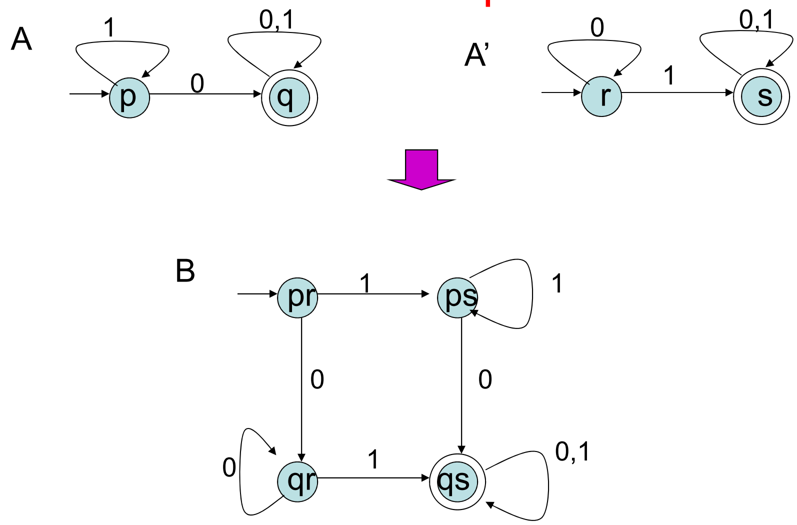
Intersection for DFA, NFA

- Suppose $A=(Q,\Sigma,\delta,q_0,F)$ is FA (DFA or NFA) for language L $A'=(Q',\Sigma,\delta',q'_0,F')$ is FA for language M.
- Product construction of an automaton B that accepts $L \cap M$: Parallel simulation of the two automata A, A'
- Define automaton $B = A \times A'$ with same alphabet Σ, set of states $Q \times Q'$, start state (q_0,q'_0), accepting set of states $F \times F'$, and transition function :

  $\delta_B((q,q'),a) = \{ (p,p') \mid p \in \delta(q,a), p' \in \delta'(q',a) \}$
- Note: If both A,A' are DFA, then B is also DFA
- Proof: Path in B from start state to $F \times F'$ labeled w $\Leftrightarrow$

  $\Leftrightarrow$ path in A to F and path in A' to F'
- Complexity: Quadratic

Example



Difference

- If L, M are regular then also L-M
- Proof: $L - M = L \cap (M^c)$

- HW: Given DFA for L and M, construct a DFA for L-M

Reversal

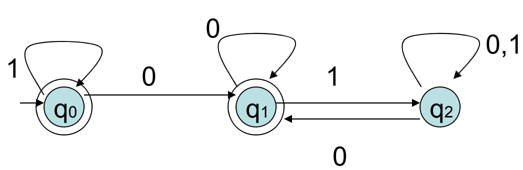
- Reversal of a string $w=a_1a_2 \ldots a_n$ is $w^R = a_n \ldots a_2a_1$
- Reversal of a language L is $L^R = \{ w^R \mid w \text{ in } L \}$
- Theorem: If L is regular then L^R is also regular.
- Proof: Via NFA. Given NFA A for L, construct ε-NFA A' for as follows:
  1. Reverse all the edges
  2. Make the start state of A be the only accepting state of A'
  3. Add a new start state and ε–transitions from it to all the accepting states of A

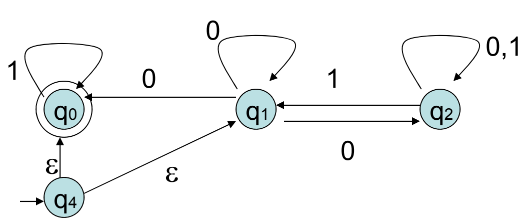
  Then, accepting path labeled w in A $\Leftrightarrow$ accepting path labeled w^R in A' (the reverse path)

HW: Give proof via REs (linear blowup); see book

Hard to do with DFAs : reversal of transitions gives NFA

Reversal example





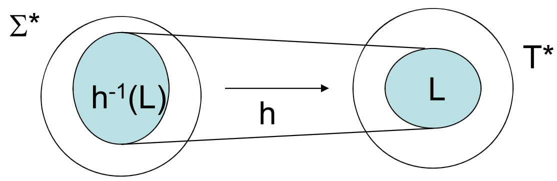
Homomorphism

- Homomorphism h: $\Sigma \rightarrow T^*$ from one alphabet Σ to an alphabet T (same or another) maps each symbol a in Σ to a string h(a) over T
- Can be extended to strings over Σ:
  $h(a_1 \ldots a_n) = h(a_1) \ldots h(a_n)$
- Example: $\Sigma=\{0,1\}$, $T=\{a,b\}$, h(0)=ab, h(1)=ε
  h(01001) = ababab
- Extended to languages: h(L) = { h(w) | w in L }

Homomorphism

- Theorem: Regular languages closed under homeomorphisms; i.e. L regular $\Rightarrow$ h(L) regular
- Proof via REs: Take a RE for L, replace every symbol a by the string h(a) $\rightarrow$ RE for h(L)
  Formal proof that it works: see book
- Example: $\Sigma=\{0,1\}$, $T=\{a,b\}$, h(0)=aa, h(1)=ε, R=0*1*
  Then h(R) = (aa)*ε* = (aa)* = set of strings with even # of a's

Inverse Homomorphism



- $h^{-1}(L) = \{ w \in \Sigma^* \mid h(w) \in L \}$
- Theorem: If L is regular then $h^{-1}(L)$ is also regular
- Proof: Via DFA. Take DFA A for L

  Construct DFA A' for $h^{-1}(L)$: same states, start state, accepting states

  Transition function of A': $\delta'(q,a) = \hat{\delta}(q,h(a))$

  Then, for any string w, $\delta'(q_0,w) \in F \Leftrightarrow \hat{\delta}(q_0,h(w)) \in F$

  $\Rightarrow$ w accepted by A' $\Leftrightarrow$ h(w) accepted by A

Use of Closure properties

- Can use closure properties to show that a language is regular (or to show that a language is not regular).

Example: L=Set of strings of length ≥ 8 that contain the substring 110 but not 100 is regular

Proof: L1={strings of length ≥ 8} regular

- L2={strings with substring 110} regular
- L3={strings with substring 100} regular
- $L = L1 \cap L2 - L3$
- Regular languages are closed under $\cap$, –

$\Rightarrow$ L is regular

Examples ctd.

- If two languages L, M are same except for finite number of strings, then they are either both regular or both nonregular
- Proof:
- N1 = L-M is finite $\Rightarrow$ regular
- N2 = M-L is finite $\Rightarrow$ regular
- $L = M \cup N1 - N2$
- $M = L \cup N2 - N1$
- Regular languages closed under $\cup$, $-$

$\Rightarrow$ If L is regular then M is regular

If M is regular then L is regular