Theory of Computation

MIEIC, 2nd Year

João M. P. Cardoso

Dep. de Engenharia Informática
Faculdade de Engenharia (FEUP)
Universidade do Porto
Porto
Portugal

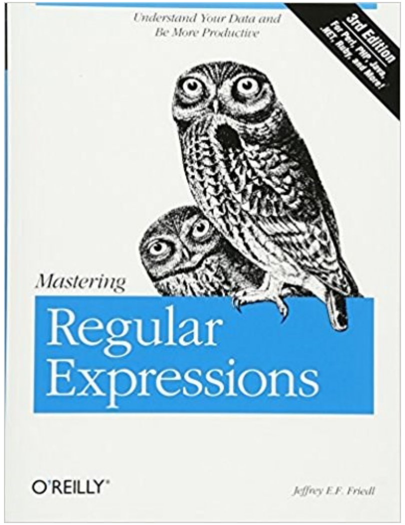
Email: jmpc@acm.org

Outline

- Regular Expressions
- Conversion of regular expressions into ε-NFAs
- Conversions of FAs into regular expressions

Regular Expressions

- Useful for searching in text (e.g., grep and sed of Linux/Unix), in compiler generators (Lex, Flex, lexical analyzers)
- Useful in applications that need to search for patterns (e.g., intrusion detection systems, anti-virus)
- Built-in in some programming languages (e.g., Perl)
  - PCRE (Perl-compatible regular expressions) format
  - POSIX regex format
- Supported by using APIs such as in Java (java.util.regex)
- And…


Understand Your Data and Be More Productive
3rd Edition
For Perl, PHP, Java, .NET, Ruby, and More!
Mastering
Regular Expressions
O'REILLY
Jeffrey E.F. Friedl

Regular Expressions

- Alternative to NFAs (including ε-NFAs) and DFAs
- Equivalent to NFAs (including ε-NFAs) and DFAs
- Algebraic characteristics allow the use of expressions to specify the strings of the language
- Regular expressions define languages
  - **Example: 01*+10***
  - L(**01*+10***): Language of the binary strings starting with a 0 followed by zero or more 1s, or starting with a 1 followed by zero or more 0s

Operators over Languages

- **Union** of two languages L and M (**L ∪ M**), is the set of the strings that belong to L, to M, or to both
  - L = {001, 10, 111} M = {ε, 001}   L ∪ M = {ε, 001, 10, 111}
- **Concatenation** of two languages L and M (**LM** or **L.M**), is the set of strings obtained by concatenating any string in L with any string in M
  - LM = {001, 10, 111, 001001, 10001, 111001}
- **Closure** of a language L (**L***) is the set of strings obtained concatenating na arbitrary number of strings of L, including repetitions, i.e., $L^* = \cup_{i\geq0} L^i$, in which $L^0=\{\varepsilon\}$
  - L= {0,1},   L* is the language of the binary strings

Closure Examples

- L = {0, 11}
  - $L^0 = \{\varepsilon\}$
  - $L^1 = L = \{0, 11\}$
  - $L^2 = LL = \{00, 011, 110, 1111\}$
  - …
  - L* = {ε , 0, 11, 00, 011, 110, 1111, …}
  - Although L is a finite language, as well as each L^i, L* is infinite
- L = {all strings with only 0s}
  - L* = L
  - L is infinite, such that L*
- L = ∅
  - $L^* = L^0 = \{\varepsilon\}$

Construction of Regular Expressions

- Basis
  - The special symbols ε e $\emptyset$ are regular expressions
    - $L(\varepsilon) = \{\varepsilon\}$ and $L(\emptyset) = \emptyset$
  - If a is a symbol, **a** is a regular expression
    - L(**a**) = {a}
  - A variable (e.g., L) is a regular expression
    - Represents any language specified by regular expressions
- Induction
  - IF E and F are regular expressions, E + F is a regular expression
    - $L(E + F) = L(E) \cup L(F)$
  - If E and F are regular expressions, EF is a regular expression
    - L(EF) = L(E)L(F)
  - If E is a regular expression, E* is a regular expression
    - L(E*) = (L(E))*
  - If E is a regular expression, (E) is a regular expression
    - L((E)) = L(E)

Regular Expressions Operators

- * (zero or more)
- . (concatenation: symbol can be omitted)
- + (or | or ∪)
- Operator precedence (from highest to lowest)
    - *
    - .
    - +
- Parenthesis can be used to “force” a certain order
- $^+$ used to represent 1 or more

Example

- Write a regular expression for the set of strings consisting of alternating 0s and 1s
  - Example: **01** L(**01**) = {01}
  - First tentative: (**01**)*
    - ≠ **01***
    - L((**01**)*) = {ε, 01, 0101, 0101, …}
    - We miss many!
  - Second tentative:
    - (**01**)*+(**10**)*+**0**(**10**)*+**1**(**01**)*
      - Right?
    - (ε+**1**)(**01**)*(ε+**0**)
      - Right?

Exercise 1

- Write regular expressions for the following languages:
  - a) the set of the strings over {a,b,c} with at least one a and at least one b
  - b) that all the pairs of adjacent 0s appear before all the pairs of adjacent 1's
- Describe the language given by the regular expression:
  - $(1+\varepsilon)(00^*1)^*0^*$

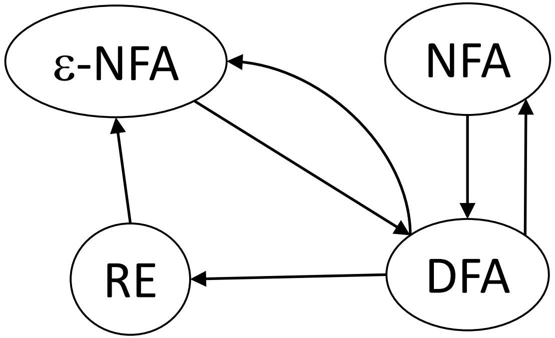
REs in UNIX/Linux

- ASCII: alphabet with 128 symbols
- Symbol ‘.’ means “any char”
- [abcd] means a+b+c+d
- [A-Z] means all the uppercase letters
- “|” means “+”
- “?” means 0 or 1 of
- “*” means 0 or more
- “+” means 1 or more
- {n} means n copies of
- Search using REs in the utility **grep** (*global regular expression and print*)

Exercise 3 (homework)

- Write a regular expression (e.g., using the PCRE format, or the format used in UNIX/Linux grep) to
  - Search for phone numbers in a text
  - Search for codex codes (with number and locality) in a text

FA – RE Equivalence



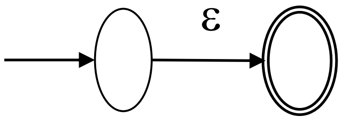
- Show that all the languages defined by FAs can be also defined by regular expressions (DFA → RE)
- Show that all the languages defined by REs can be also defined by FAs (RE → ε-NFA)

From Res to FAs

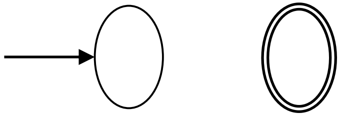
From REs to FAs

- Theorem: every language defined by a regular expression is also defined by an FA.
- Proof: structural induction over the definition of the regular expression
  - Basis step: ε, ∅ and a
  - Induction step: union, concatenation and closure
  - L = L(R) = L(E), E is a ε-NFA with
    - Exactly one accept state
    - Without input transitions to the start state
    - Without output transitions from the final state

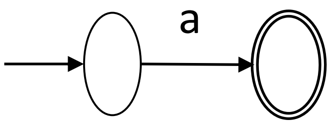
Basis Step



Empty String

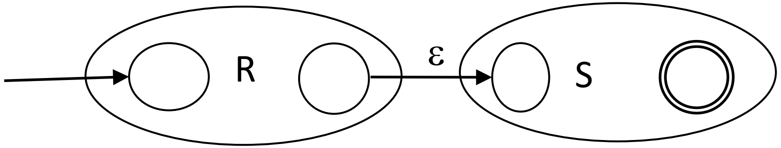


∅

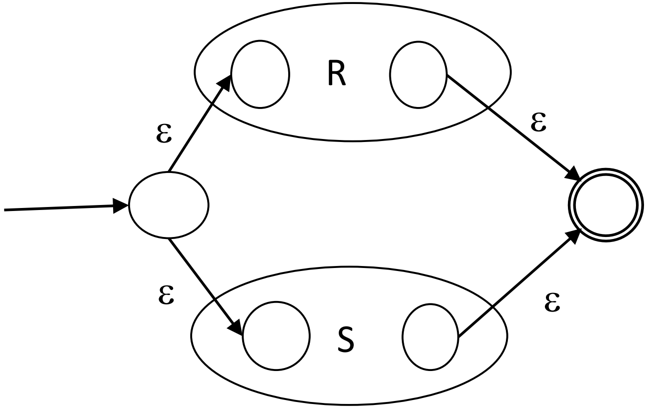


symbol

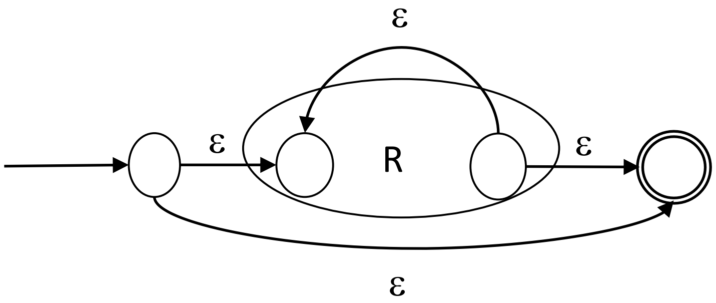
Induction Step



Concatenation: RS



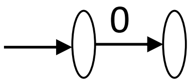
Union: R+S

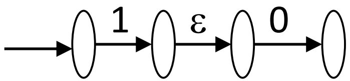


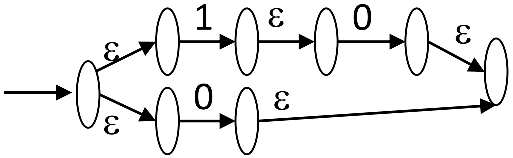
- Closure: R*

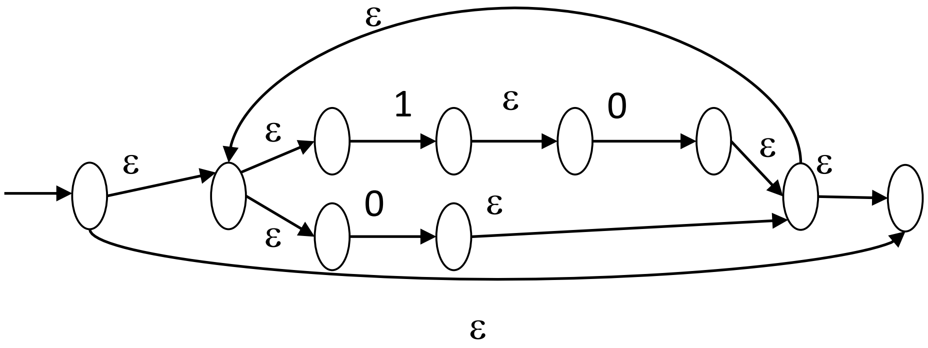
Example

- Draw an ε-NFA for the RE (0+10)*11(0+1)*
  - Try to simplify the FA

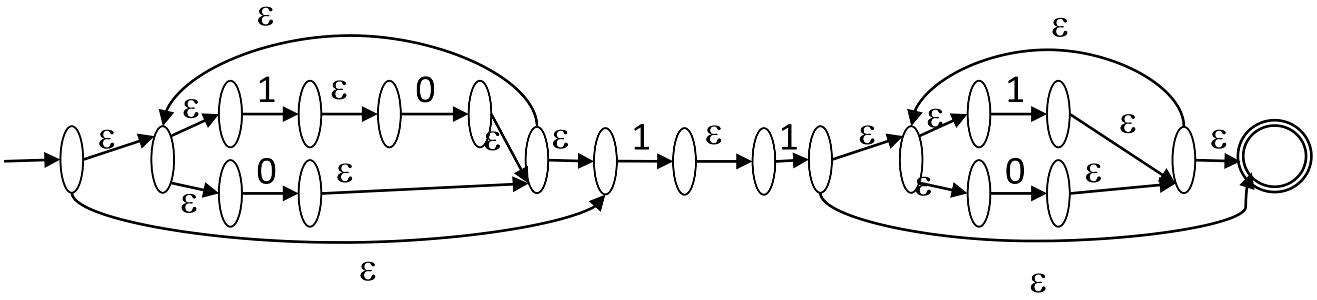


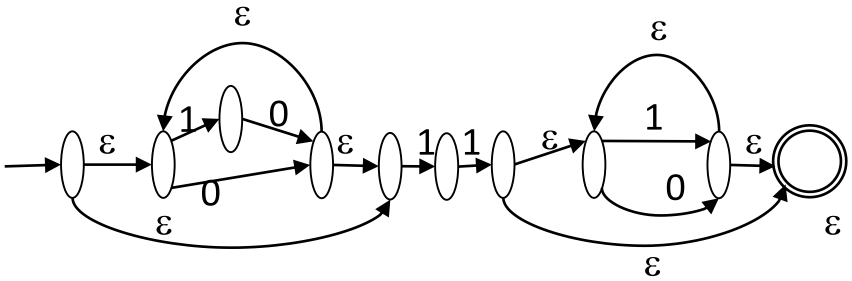


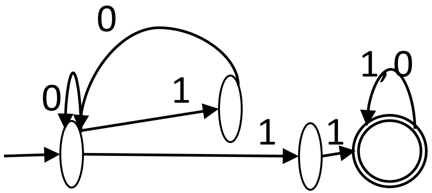




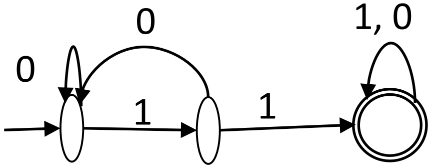
Example (cont.)







- Simplification



From DFAs to REs

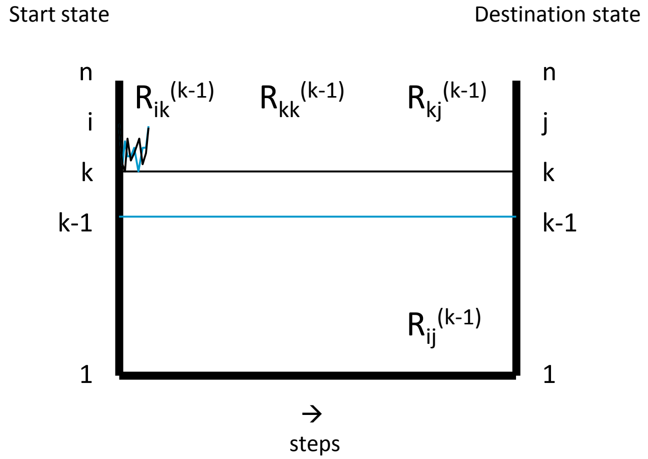
- **Theorem**: If L=L(A) for a DFA A then it exists a regular expression R such that L=L(R)
- Two methods:
  - **Construction of Paths:** Enumerate the states from 1 to n; build the REs that successively describe **paths** more complex in the DFA, until they describe all the paths from the start state to each accept state
  - **State Elimination:** Consider the transitions labeled by REs; **eliminate** the internal states substituting their “behavior” by REs

FAs to REs

Construction of Paths

Construction of Paths

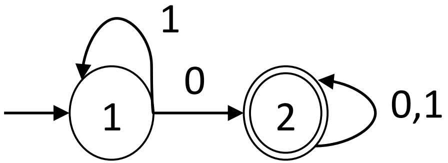
- Numerate the nodes (states) from 1 to n
- $R_{ij}^{(k)}$
  - Regular expression defining the language consisting of the set of strings such that w is the label of a path between nodes i and j, without passing in any intermediate node higher than k
- Induction in the number of nodes (k)



Construction of Paths

- Basis
  - k=0 means without intermediate nodes (the lowest is 1)
    - Edge from i to j (RE is the respective symbol; or $\emptyset$, if does not exist; or $a_1+a_2+\ldots+a_m$, if there are m edges)
    - Node i (i to i) (RE is $\varepsilon+a_1+a_2+\ldots+a_m$)
- Induction
  - Hypothesis: the paths that use nodes until k-1 are already converted
  - Exist a path from i to j without passing in node k
    - $R_{ij}^{(k-1)}$
  - The path passes one or more times in k:
    - $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$
  - End: $R_{ij}^{(n)}$ paths between i and j considering all the nodes
- The RE of the language of the FA is the union of the regular expressions $R_{1j}^{(n)}$ such that j is na accept state.

Example DFA $\Rightarrow$ RE



- FA that recognizes strings with at least one 0
- $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$
- $R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)} (R_{11}^{(0)})^* R_{1j}^{(0)}$

$R_{11}^{(0)}$	$\varepsilon+1$
$R_{12}^{(0)}$	0
$R_{21}^{(0)}$	$\varnothing$
$R_{22}^{(0)}$	$\varepsilon+0+1$

$R_{11}^{(1)}$	$\varepsilon+1+(\varepsilon+1)(\varepsilon+1)^*(\varepsilon+1)$	1^*
$R_{12}^{(1)}$	$0+(\varepsilon+1)(\varepsilon+1)^*0$	1^*0
$R_{21}^{(1)}$	$\varnothing+\varnothing(\varepsilon+1)^*(\varepsilon+1)$	$\varnothing$
$R_{22}^{(1)}$	$\varepsilon+0+1+\varnothing(\varepsilon+1)^*0$	$\varepsilon+0+1$

Simplification:

$(\varepsilon+1)^* = 1^*$

$\varnothing R = R\varnothing = \varnothing$

$\varnothing+R = R+\varnothing = R$

$R_{11}^{(2)}$	$1^* + 1^*0(\varepsilon+0+1)^*\varnothing$	1^*
$R_{12}^{(2)}$	$1^*0 + 1^*0(\varepsilon+0+1)^*(\varepsilon+0+1)$	$1^*0(0+1)^*$
$R_{21}^{(2)}$	$\varnothing + (\varepsilon+0+1)(\varepsilon+0+1)^*\varnothing$	$\varnothing$
$R_{22}^{(2)}$	$\varepsilon+0+1+(\varepsilon+0+1)(\varepsilon+0+1)^*(\varepsilon+0+1)$	$(0+1)^*$

$R = 1^*0(0+1)^*$

FAs to REs

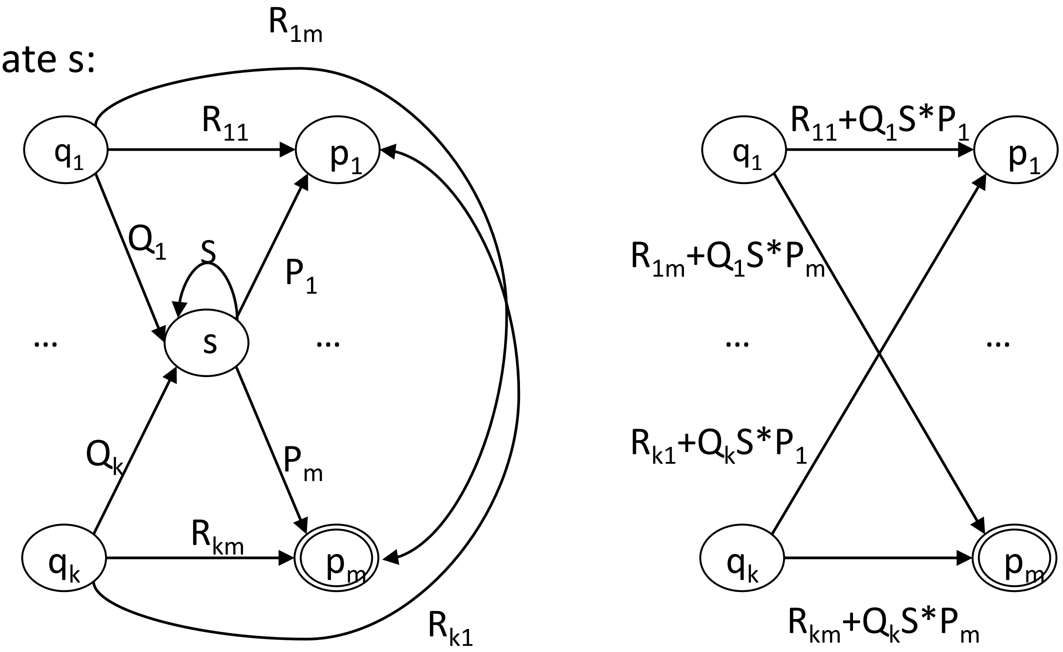
State Elimination

State Elimination

- The construction of paths has many repetitions, is onerous, and may provide long/complex REs if we don't simplify them
- State elimination technique
  - Build REs representing all the implicit strings in the part of the diagram we are substituting
  - Simplify the diagram making more complex the labels of the edges that remain
- State to eliminate: s
  - States q_i include all the source states of s
  - States p_j include all the sink states of s (they can intersect the states q_i)
  - Remove s and all the edges that connect s, adding in all the edges from q_i to p_j a part of the eventual path from q_i to p_j, over s, including the cycle in s: $Q_iS^*P_j$

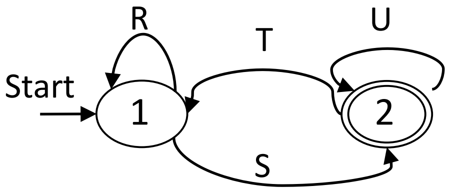
State Elimination

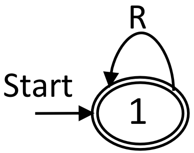
- Eliminating state s:



Strategy

- Eliminate the intermediate states, maintaining the start and the accept states, until you have a single edge with the respective RE from the start to each accept state
- Union of the alternatives
- If q≠q0, we obtain an FA with 2 states
  - (R+SU*T)*SU*
- If not we obtain na FA with a single state
  - R*

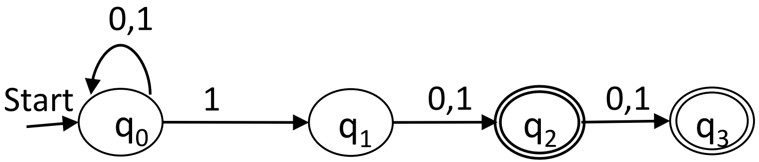


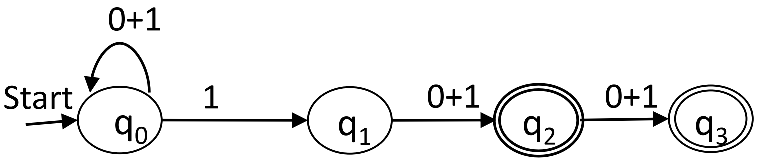


Example

- ➊Start by substituting the labels in the transitions to REs
- ➋Successively eliminate the nodes that are neither start nor accept and substitute each node eliminated by the respective RE
- Note: consider one FA for each accept state and the final RE is obtained by the union of the individual REs (one per FA)

➊



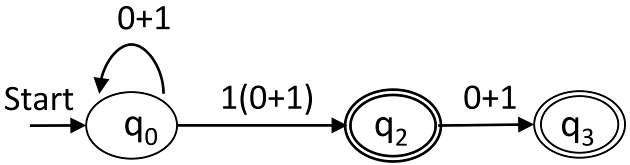


- Q_1= 1, P_1 = 0+1, R_{02}= ∅, S = ∅
- New edge q_0-q_2: ∅ + 1∅*(0+1) = 1(0+1)
  - Since: L(∅*)={ε} ∪ L(∅) ∪ L(∅) L(∅)…

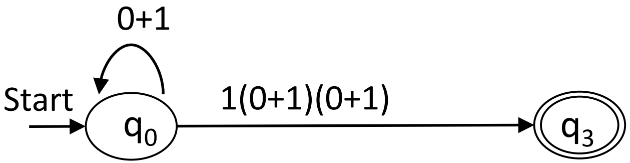
Example (cont.)

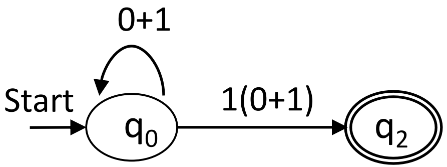
➊Start by substituting the labels in the transitions to REs
➋Successively eliminate the nodes that are neither start nor accept and substitute each node eliminated by the respective RE

➋



Consider one FA for each accept state:



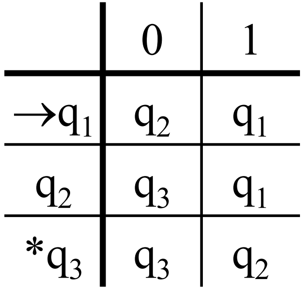


The final RE is obtained by the union of the individual REs (one per FA):

- RE = $(0+1)^*1(0+1)(0+1) + (0+1)^*1(0+1) = (0+1)^*1(0+1)(\varepsilon+0+1)$

Exercise 2

- Consider a DFA with the transition table given below
  1. Calculate all the expressions $R_{ij}^{(0)}$ (i is the number of state q_i)
  2. Calculate all the expressions $R_{ij}^{(1)}$ and simplify them
  3. Obtain a regular expression for the language of the DFA
  4. Draw the transition (state) diagram of the DFA and obtain a regular expression for its language using the state elimination technique



	0	1
$\rightarrow q_1$	q_2	q_1
q_2	q_3	q_1
$*q_3$	q_3	q_2

Algebraic Rules for REs

Algebraic Rules for REs

- Two REs are equivalent if they define the same language
  - Two REs with variables are equivalent if, whatever the languages substituting the variables, both REs define the same language
- Main interest: simplify REs
- Commutativity
  - Union: L + M = M + L
  - Concatenation: does not exist!
- Associativity
  - Union: (L + M) + N = L + (N + M)
  - Concatenation: (LM)N = L(MN)

Algebraic Laws for REs (cont.)

- Identity
  - Union: $\emptyset + L = L + \emptyset = L$
  - Concatenation: $\varepsilon L = L\varepsilon = L$
- Absorption
  - Concatenation: $L\emptyset = \emptyset L = \emptyset$
  - Union: does not exist
- Distributive
  - Of the concatenation over the union
  - left: $L(M + N) = LM + LN$
  - right: $(M + N)L = ML + NL$

Algebraic Laws for REs (cont.)

- Idempotent
  - Union: $L + L = L$
  - Concatenation: don’t exist
- Example: simplify $0 + 01*$
  - $0 + 01* =$
  - $0\varepsilon + 01* =$ identity of the concatenation
  - $0(\varepsilon + 1*) =$ distributive of the concatenation over the union
  - $01*$ because ε belongs to the language $1*$

Exercise 4

- Let’s use the previous example:
  - RE = (0+1)*1(0+1)(0+1) + (0+1)*1(0+1) = (0+1)*1(0+1)(ε+0+1)
- How to simplify using RE rules?
  - (0+1)*1(0+1)(0+1) + (0+1)*1(0+1)
  - (0+1)*(1(0+1)(0+1) + 1(0+1)) [rule?]
  - (0+1)*1((0+1)(0+1) + (0+1)) [rule?]
  - (0+1)*1(0+1)((0+1) + ε) [rule?]
  - (0+1)*1(0+1)(ε+0+1) [rule?]

Algebraic Laws Involving Closure

- $(L^*)^* = L^*$
- $\varnothing^* = \varepsilon$ Why?
- $\varepsilon^* = \varepsilon$
- $L^+ = LL^* = L^*L$
- $L^* = L^+ + \varepsilon$
- $L? = \varepsilon + L$

- Exercise: in which conditions we obtain $L^* = L^+$?

Discovering New Laws

- Example: $(L + M)^* = (L^*M^*)^*$ is a law?
- Proof: →
  - Supposing $w \in (L+M)^*$
  - $w = w_1w_2\ldots w_k$, where $w_i \in L$ or $w_i \in M$
  - Then w_i is also in L^*M^*, because if it is in L then it is also in L^* and if we take $M^* = \varepsilon$ …
  - Needed to prove ← **(TPC)**
- Alternative: transform the expression in a concrete RE and analyze the languages
  - $(L + M)^*$ can be transformed in the concrete $(a+b)^*$ and $(L^*M^*)^*$ to $(a^*b^*)^*$
  - In both cases we conclude that $L(E) = \Sigma^*$

Test for Algebraic Laws

- To test if E = F, where E and F are REs with the same set of variables
  - Convert E and F in the concrete REs C and D, substituting each variable by a symbol
  - Test if L(C) = L(D); if true then E=F is a law, else if not a law.
- Examples:
  - Is L* = L*L* ?
    - Converting: C=a* and D=a*a*; both are the set of all strings over {a}
    - Then L(C) = L(D) and "the concatenation of a closure language with itself produces the same language" is a law
  - Is L + ML = (L+M)L ?
    - C= a+ba, D= (a+b)a = aa + ba then L(C)≠L(D) and the statement is not a law

Limits of the Test

- The test becomes invalid if we consider other operators than the ones of the REs
- Example: add the interception operator to the algebra of the REs
  - Note: the ∩ operator does not empower the language (the languages we can define are the same)
  - Is L ∩ M ∩ N = L ∩ M ? The interception of 3 is the same as 2? Obviously false, but:
    - Substituting L=a, M=b, N=c we get {a} ∩ {b} ∩ {c} = {a} ∩ {b} = ∅ and the test would give true.

Exercise 5

- Proof or give a counter-example for the following:
  - $(R+S)^* = R^* + S^*$
  - $(RS+R)^*R = R(SR+R)^*$ (homework)
  - $(RS+R)^*RS = (RR^*S)^*$

Conclusions

Conclusions

- Regular Expressions (REs) provide a way to specify languages (named as regular languages)
- REs can be converted in ε-NFAs
- FAs can be converted into REs