Finite Automata and Formal Languages

TMV027/DIT321– LP4 2013

Lecture 7
Ana Bove

April 18th 2013

Overview of today's lecture:

- More on RE;
- Algebraic laws for regular expressions;
- Equivalence between FA and RE: from FA to RE.

Recall: RE and the Language they Define

$$R, S ::= \emptyset \mid \epsilon \mid a \mid R + S \mid RS \mid R^*$$

Definition: The *language* defined by a regular expression is defined by recursion on the expression:

Base cases:
- $\mathcal{L}(\emptyset) = \emptyset$;
- $\mathcal{L}(\epsilon) = \{\epsilon\}$;
- Given $a \in \Sigma$, $\mathcal{L}(a) = \{a\}$.

Recursive cases:
- $\mathcal{L}(R + S) = \mathcal{L}(R) \cup \mathcal{L}(S)$;
- $\mathcal{L}(RS) = \mathcal{L}(R)\mathcal{L}(S)$;
- $\mathcal{L}(R^*) = \mathcal{L}(R)^*$.

Example of Regular Expressions

Let $\Sigma = \{0, 1\}$:

- $(01)^* = \{\epsilon, 01, 0101, 010101, \ldots\}$
- $0^* + 1^* = \{\epsilon, 0, 00, 000, \ldots\} \cup \{\epsilon, 1, 11, 111, \ldots\}$
- $(0 + 1)^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots\}$
- $(000)^* = \{\epsilon, 000, 000000, 000000000, \ldots\}$
- $01^* + 1 = \{0, 01, 011, 0111, \ldots\} \cup \{ 1\}$
- $((0(1^*)) + 1) = \{0, 01, 011, 0111, \ldots\} \cup \{ 1\}$
- $(01)^* + 1 = \{\epsilon, 01, 0101, 010101, \ldots\} \cup \{ 1\}$
- $(\epsilon + 1)(01)^*(\epsilon + 0) = (01)^* + 1(01)^* + (01)^*0 + 1(01)^*0$
- $(01)^* + 1(01)^* + (01)^*0 + 1(01)^*0$

What do they mean? Are there expressions that are equivalent?

Algebraic Laws for Regular Expressions

The following equalities hold for any RE R, S and T:

Idempotent:	$R + R = R$	
Commutative:	$R + S = S + R$	In general, $RS \neq SR$
Associative:	$R + (S + T) = (R + S) + T$	$R(ST) = (RS)T$
Distributive:	$R(S + T) = RS + RT$	$(S + T)R = SR + TR$
Identity:	$R + \emptyset = \emptyset + R = R$	$R\epsilon = \epsilon R = R$
Annihilator:	$R\emptyset = \emptyset R = \emptyset$	
	$\emptyset^* = \epsilon^* = \epsilon$	
	$R? = \epsilon + R$	
	$R^+ = RR^* = R^*R$	
	$R^* = (R^*)^* = R^*R^* = \epsilon + R^+$	

Note: Compare these laws with those for sets on slide 14 lecture 2.

Algebraic Laws for Regular Expressions

Other useful laws to simplify regular expressions are:

- *Shifting rule:* $R(SR)^* = (RS)^*R$
- *Denesting rule:* $(R^*S)^*R^* = (R+S)^*$

  Note: By the shifting rule we also get $R^*(SR^*)^* = (R+S)^*$
- Variation of the denesting rule: $(R^*S)^* = \epsilon + (R+S)^*S$

Example: Proving Equalities Using the Algebraic Laws

Example: A proof that $a^*b(c+da^*b)^* = (a+bc^*d)^*bc^*$:

$a^*b(c+da^*b)^* = a^*b(c^*da^*b)^*c^*$		by denesting $(R = c, S = da^*b)$
$a^*b(c^*da^*b)^*c^* = (a^*bc^*d)^*a^*bc^*$		by shifting $(R = a^*b, S = c^*d)$
$(a^*bc^*d)^*a^*bc^* = (a+bc^*d)^*bc^*$		by denesting $(R = a, S = bc^*d)$

Example: The set of all words with no substring of more than two adjacent 0's is $(1+01+001)^*(\epsilon+0+00)$. Now,

$$\begin{aligned}(1+01+001)^*(\epsilon+0+00) &= ((\epsilon+0)(\epsilon+0)1)^*(\epsilon+0)(\epsilon+0)\\ &= (\epsilon+0)(\epsilon+0)(1(\epsilon+0)(\epsilon+0))^* \qquad \text{by shifting}\\ &= (\epsilon+0+00)(1+10+100)^*\end{aligned}$$

Then $(1+01+001)^*(\epsilon+0+00) = (\epsilon+0+00)(1+10+100)^*$

Equality of Regular Expressions

Remember that RE are a way to denote languages.

Then, for RE R and S, $R = S$ actually means $\mathcal{L}(R) = \mathcal{L}(S)$.

Hence we can prove the equality of RE in the same way we can prove the equality of languages.

Example: Let us prove that $R^* = R^*R^*$. Let $\mathcal{L} = \mathcal{L}(R)$.

$\mathcal{L}^* \subseteq \mathcal{L}^*\mathcal{L}^*$ since $\epsilon \in \mathcal{L}^*$.

Conversely, if $\mathcal{L}^*\mathcal{L}^* \subseteq \mathcal{L}^*$ then $x = x_1x_2$ with $x_1 \in \mathcal{L}^*$ and $x_2 \in \mathcal{L}^*$.

If $x_1 = \epsilon$ or $x_2 = \epsilon$ then it is clear that $x \in \mathcal{L}^*$.

Otherwise $x_1 = u_1u_2 \ldots u_n$ with $u_i \in \mathcal{L}$ and $x_2 = v_1v_2 \ldots v_m$ with $v_j \in \mathcal{L}$.

Then $x = x_1x_2 = u_1u_2 \ldots u_nv_1v_2 \ldots v_m$ is in $\mathcal{L}^*$.

Proving Algebraic Laws for Regular Expressions

In general, given the RE R and S we can prove the law $R = S$ as follows:

1. Convert R and S into *concrete* regular expressions C and D, respectively, by replacing each variable in the RE R and S by (different) concrete symbols.

   Example: $R(SR)^* = (RS)^*R$ can be converted into $a(ba)^* = (ab)^*a$.

2. Prove or disprove whether $\mathcal{L}(C) = \mathcal{L}(D)$. If $\mathcal{L}(C) = \mathcal{L}(D)$ then $R = S$ is a true law, otherwise it is not.

Theorem: *The above procedure correctly identifies the true laws for RE.*

Proof: See theorems 3.14 and 3.13 in pages 121 and 120 respectively.

Example: Proving the shifting law was (somehow) one of the exercises in assignment 1: prove that for all n, $a(ba)^n = (ab)^na$.

Example: Proving the Denesting Rule

We can state $(R^*S)^*R^* = (R+S)^*$ by proving $\mathcal{L}((a^*b)^*a^*) = \mathcal{L}((a+b)^*)$:

$\subseteq$: Let $x \in (a^*b)^*a^*$, then $x = vw$ with $v \in (a^*b)^*$ and $w \in a^*$.

By induction on v. If $v = \epsilon$ we are done.

Otherwise $v = av'$ or $v = bv'$.
In both cases $v' \in (a^*b)^*$ hence by IH $v'w \in (a+b)^*$ and so is vw.

$\supseteq$: Let $x \in (a+b)^*$.

By induction on x. If $x = \epsilon$ then we are done.

Otherwise $x = x'a$ or $x = x'b$ and $x' \in (a+b)^*$.

By IH $x' \in (a^*b)^*a^*$ and then $x' = vw$ with $v \in (a^*b)^*$ and $w \in a^*$.

If $x'a = v(wa) \in (a^*b)^*a^*$ since $v \in (a^*b)^*$ and $(wa) \in a^*$.
If $x'b = (v(wb))\epsilon \in (a^*b)^*a^*$ since $v(wb) \in (a^*b)^*$ and $\epsilon \in a^*$.

Regular Languages and Regular Expressions

Theorem: *If $\mathcal{L}$ is a regular language then there exists a regular expression R such that $\mathcal{L} = \mathcal{L}(R)$.*

Proof: Recall that each regular language has an automata that recognises it.

We shall construct a regular expression from such automata.

We will see 2 ways of constructing a regular expression from an automata.

- Eliminating states (section 3.2.2);
- By solving a *linear equation system* using Arden's Lemma (**OBS:** not in the book!)

From FA to RE: Eliminating States in an Automaton A

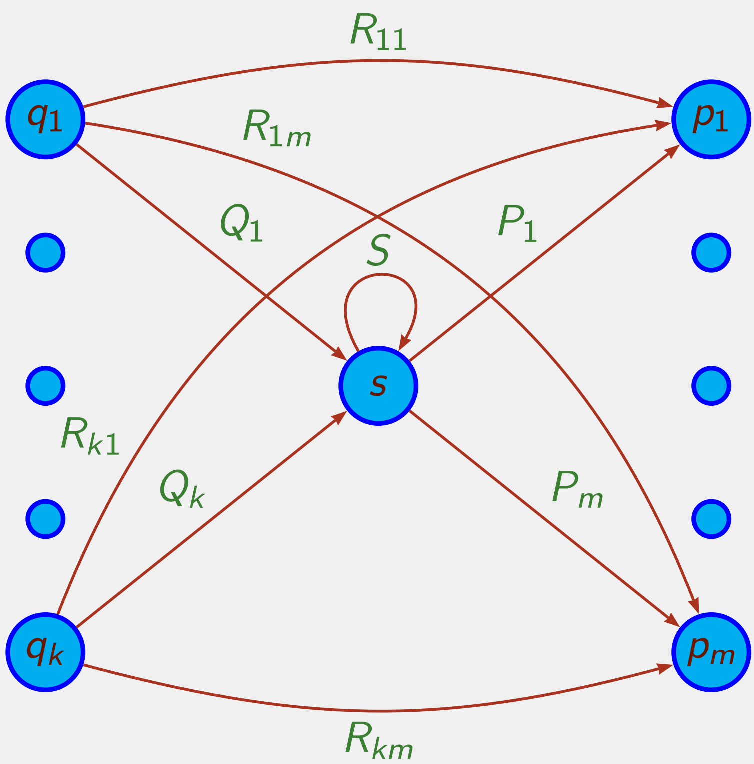
This method of constructing a RE from a FA involves eliminating states.

When we eliminate the state s, all the paths that went through s do not longer exists!

To preserve the language of the automaton we must include, on an arc that goes directly from q to p, the labels of the paths that went from q to p passing through s.

Labels now are not just symbols but (possible an infinite number of) strings: hence we will use RE as labels.

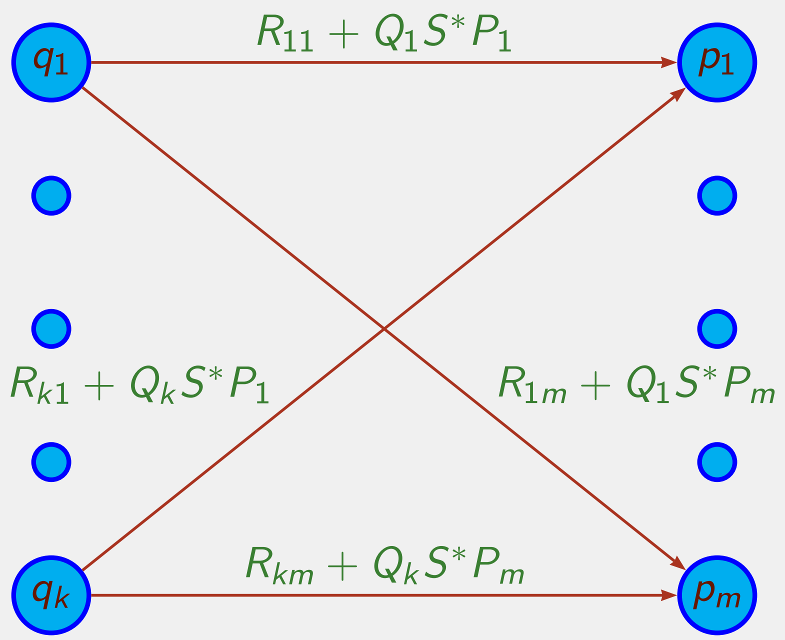
Eliminating State s in A



If an arc does not exist in A, then it is labelled $\emptyset$ here.

For simplification, we assume the q's are different from the p's.

Eliminating State s in A

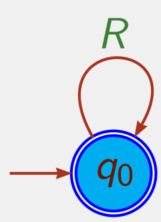


Eliminating States in A

For *each accepting* state q we proceed as before until we have only q_0 and q left.

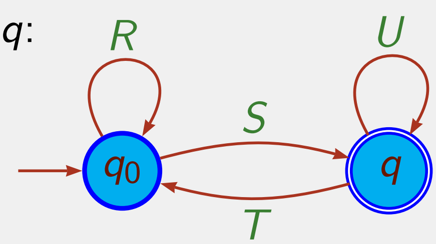
For each accepting state q we have 2 cases: $q_0 = q$ or $q_0 \neq q$.

If $q_0 = q$:



The expression is R^*.

If $q_0 \neq q$:

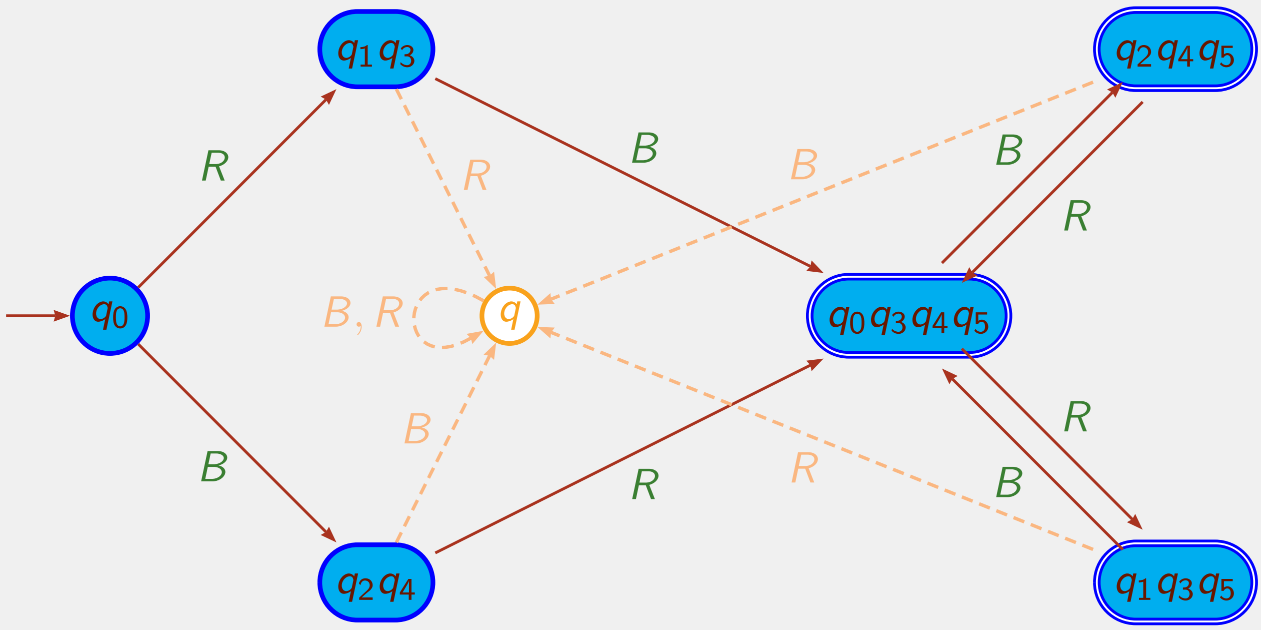


The expression is $(R + SU^*T)^*SU^*$.

The final expression is the sum of the expressions derived for each final state.

Example: Regular Expression Representing Gilbreath's Principle

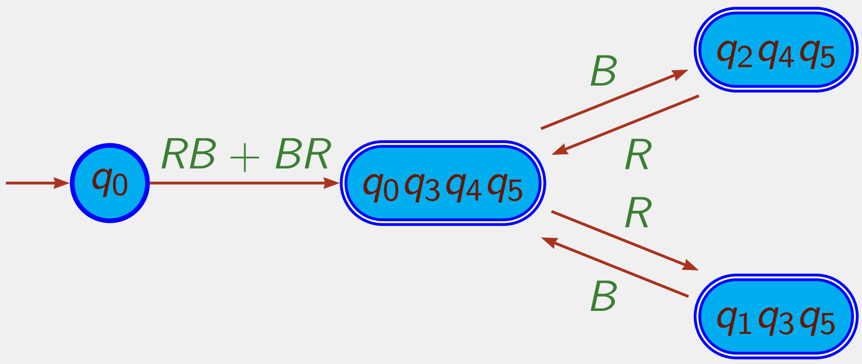
Recall:



Observe: Eliminating q is trivial. Eliminating q_1q_3 and q_2q_4 is also easy.

Example: Regular Expression Representing Gilbreath's Principle

After eliminating q, q_1q_3 and q_2q_4 we get:



- RE when final state is $q_0q_3q_4q_5$:
  $(RB + BR)(RB + BR)^* = (RB + BR)^+$
- RE when final state is $q_2q_4q_5$: $(RB + BR)(RB)^*B(R(RB)^*B)^*$
- RE when final state is $q_1q_3q_5$: $(RB + BR)(BR)^*R(B(BR)^*R)^*$

Example: Regular Expression Representing Gilbreath's Principle

The final RE is the sum of the 3 previous expressions.

Let us first do some simplifications.

$(RB+BR)(RB)^*B(R(RB)^*B)^* = (RB+BR)(RB)^*(BR(RB)^*)^*B$ by shifting

$= (RB+BR)(RB+BR)^*B$ by the shifted-denesting rule

$= (RB+BR)^+B$

Similarly $(RB+BR)(BR)^*R(B(BR)^*R)^* = (RB+BR)^+R$.

Hence the final RE is

$$(RB+BR)^+ + (RB+BR)^+B + (RB+BR)^+R$$

which is equivalent to

$$(RB+BR)^+(\epsilon+B+R)$$

From FA to RE: Linear Equation System

To any automaton we associate a system of equations such that the solution will be REs.

At the end we get a RE for the language recognised by the automaton.

This works for DFA, NFA and ϵ-NFA.

To every state q_i we associate a variable E_i.

Each E_i represents the set $\{x \in \Sigma^* \mid \hat{\delta}(q_i, x) \in F\}$ (for DFA).

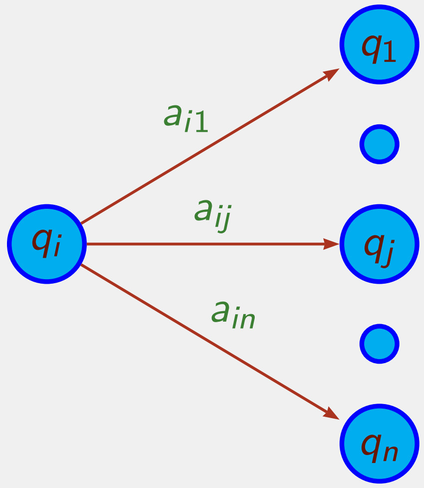
Then E_0 represents the set of words accepted by the FA.

The solution to the linear system of equations associates a RE to each variable E_i.

Then the solution for E_0 is the RE generating the same language that is accepted by the FA.

Constructing the Linear Equation System

Consider a state q_i and all the transactions coming out if it:



Then we have the equation

$$E_i = a_{i1}E_1 + \ldots + a_{ij}E_j + \ldots + a_{in}E_n$$

If q_i is final then we add ϵ

$$E_i = \epsilon + a_{i1}E_1 + \ldots + a_{ij}E_j + \ldots + a_{in}E_n$$

If there is no arrow coming out of q_i

then $E_i = \emptyset$ if q_i is not final

or $E_i = \epsilon$ if q_i is final

Solving the Linear Equation System

Lemma: *(Arden) A solution to $X = RX + S$ is $X = R^*S$. Furthermore, if $\epsilon \notin \mathcal{L}(R)$ then this is the only solution to the equation $X = RX + S$.*

Proof: We have that $R^* = RR^* + \epsilon$.

Hence $R^*S = RR^*S + S$ and then $X = R^*S$ is a solution to $X = RX + S$.

One should also prove that:

- Any solution to $X = RX + S$ contains at least R^*S;
- If $\epsilon \notin \mathcal{L}(R)$ then R^*S is the only solution to the equation $X = RX + S$ (that is, no solution is "bigger" than R^*S).

Note: See for example Theorem 6.1, pages 185–186 of *Theory of Finite Automata, with an introduction to formal languages* by John Carroll and Darrell Long, Prentice-Hall International Editions.

Example: Regular Expression Representing Gilbreath's Principle

We obtain the following system of equations (see slide 14):

$$
\begin{aligned}
E_0 &= 1E_{13} + 0E_{24} & E_{0345} &= \epsilon + 0E_{245} + 1E_{135} \\
E_{13} &= 0E_{0345} + 1E_q & E_{245} &= \epsilon + 1E_{0345} \\
E_{24} &= 1E_{0345} + 0E_q & E_{135} &= \epsilon + 0E_{0345} \\
& & E_q &= \emptyset
\end{aligned}
$$

This can be simplified to:

$$
\begin{aligned}
E_0 &= 1E_{13} + 0E_{24} & E_{0345} &= \epsilon + 0E_{245} + 1E_{135} \\
E_{13} &= 0E_{0345} & E_{245} &= \epsilon + 1E_{0345} \\
E_{24} &= 1E_{0345} & E_{135} &= \epsilon + 0E_{0345}
\end{aligned}
$$

Example: Regular Expression Representing Gilbreath's Principle

And further to:

$$
\begin{aligned}
E_0 &= (10 + 01)E_{0345} \\
E_{0345} &= (10 + 01)E_{0345} + \epsilon + 0 + 1
\end{aligned}
$$

Then a solution to E_{0345} is

$$(10 + 01)^*(\epsilon + 0 + 1)$$

and the RE which is the solution to the problem is

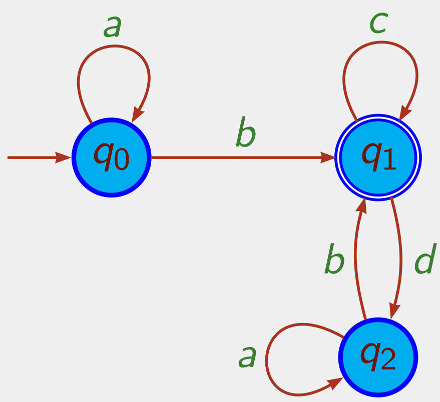
$$(10 + 01)(10 + 01)^*(\epsilon + 0 + 1)$$

or

$$(10 + 01)^+(\epsilon + 0 + 1)$$

Example: Eliminating States

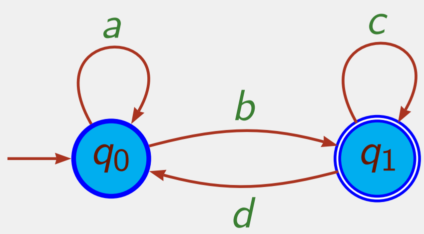
Consider the automaton D



By eliminating states the expression is

$$a^* b(c + da^* b)^*$$

Consider the automaton D'



By eliminating states the expression is

$$(a + bc^* d)^* bc^*$$

Example: Linear Equation System

The linear equations corresponding to the automaton D' are

$$E_0 = aE_0 + bE_1 \qquad E_1 = \epsilon + cE_1 + dE_0$$

The resulting RE depends on the order we solve the system.

If we eliminate E_1 first we get $E_0 = (a + bc^* d)^* bc^*$.

If we eliminate E_0 first we get $E_0 = a^* b(c + da^* b)^*$.

It should then be that $a^* b(c + da^* b)^* = (a + bc^* d)^* bc^*$!

(See the proof in slide 5.)

What RE do we obtain for the automaton D?

Overview of Next Lecture

Guest lecture by *Wolfgang Ahrendt*

Büchi automata and their application to software verification.

and sections 3.2.3, 4–4.2.1:

- Equivalence between FA and RE: from RE to FA;
- Pumping Lemma for RL;
- Closure properties of RL.

Please revise concepts in lecture 2.