Finite Automata Theory and Formal Languages

TMV027/DIT321– LP4 2016

Lecture 7
Ana Bove

April 21st 2016

Overview of today's lecture:

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

Recap: Non-deterministic Finite Automata (with ϵ-Transitions)

- Product of NFA as for DFA, accepting intersection of languages;
- Union of languages comes naturally, complement not so "immediate";
- By allowing ϵ-transitions we obtain ϵ-NFA:
  - Defined by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$;
  - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}ow(Q)$;
  - ECLOSE needed for $\hat{\delta}$;
  - Accept set of words x such that $\hat{\delta}(q_0, x) \cap F \neq \emptyset$;
  - Given a ϵ-NFA E we can convert it to a DFA D such that $\mathcal{L}(E) = \mathcal{L}(D)$;
  - Hence, also accept the so called regular language.

Regular Expressions

Regular expressions (RE) are an "algebraic" way to denote languages.

RE are a simple way to express the strings in a language.

Example: `grep` command in UNIX (K. Thompson) takes a (variation) of a RE as input.

We will show that RE are as expressive as DFA and hence, they define all and only the *regular languages*.

Inductive Definition of Regular Expressions

Definition: Given an alphabet Σ, we inductively define the *regular expressions* over Σ as follows:

Base cases:
- The constants $\emptyset$ and ϵ are RE;
- If $a \in \Sigma$ then a is a RE.

Inductive steps: Given the RE R and S, we define the following RE:
- $R + S$ and RS are RE;
- R^* is RE.

The precedence of the operands is the following:

- The closure operator * has the highest precedence;
- Next comes concatenation;
- Finally, comes the operator $+$;
- We use parentheses (,) to change the precedence.

(Compare with exponentiation, multiplication and addition on numbers.)

Another Way to Define the Regular Expressions

A nicer way to define the regular expressions is by giving the following BNF (Backus-Naur Form), for $a \in \Sigma$:

$$R ::= \emptyset \mid \epsilon \mid a \mid R + R \mid RR \mid R^*$$

alternatively

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

Note: BNF is a way to declare the syntax of a language.

It is very useful when describing *context-free grammars* and in particular the syntax of (big parts of) most programming languages.

Functional Representation of Regular Expressions

```
data RExp a = Empty | Epsilon | Atom a |
              Plus (RExp a) (RExp a) |
              Concat (RExp a) (RExp a) |
              Star (RExp a)
```

For example the expression $b + (bc)^*$ is given as

```
Plus (Atom "b") (Star (Concat (Atom "b") (Atom "c")))
```

Language Defined by the Regular Expressions

Definition: Given a RE R, the *language* $\mathcal{L}(R)$ generated/defined by it 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:
- $\mathcal{L}(R + S) = \mathcal{L}(R) \cup \mathcal{L}(S)$;
- $\mathcal{L}(RS) = \mathcal{L}(R)\mathcal{L}(S)$;
- $\mathcal{L}(R^*) = \mathcal{L}(R)^*$.

Note: $x \in \mathcal{L}(R)$ iff x is generated by R.

Notation: We write $x \in R$ or $x \in \mathcal{L}(R)$ indistinctly.

Example of Regular Expressions

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

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

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 (some of) 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: $(R^*S)^* = \epsilon + (R + S)^*S$

Note: Not always trivial to apply these rules...

Example: Proving Equalities Using the Algebraic Laws

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

$$a^*b(c + da^*b)^* = a^*b(c^*da^*b)^*c^* \quad \text{by denesting } (R = c, S = da^*b)$$
$$a^*b(c^*da^*b)^*c^* = (a^*bc^*d)^*a^*bc^* \quad \text{by shifting } (R = a^*b, S = c^*d)$$
$$(a^*bc^*d)^*a^*bc^* = (a + bc^*d)^*bc^* \quad \text{by denesting } (R = a, S = bc^*d)$$

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

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

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

Equality of Regular Expressions

Recall: 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 show 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.

   Example: We can prove the shifting law by induction: $\forall n \in \mathbb{N}.a(ba)^n = (ab)^n a$.

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 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 RE R such that $\mathcal{L} = \mathcal{L}(R)$.*

Proof: Recall that each regular language has a FA that recognises it.

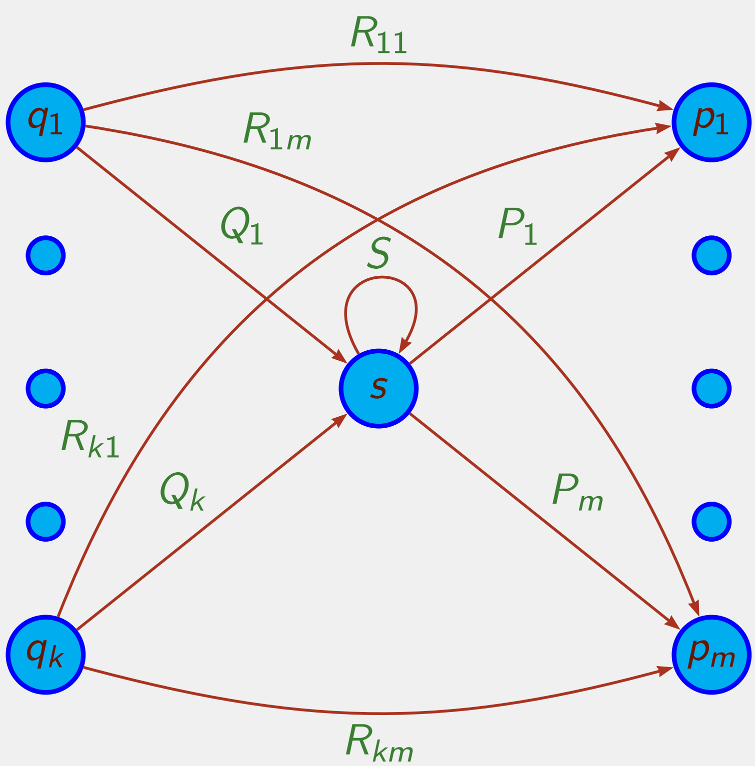
We shall construct a RE from such automaton.

We shall see 2 ways of constructing a RE from a FA:

- 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

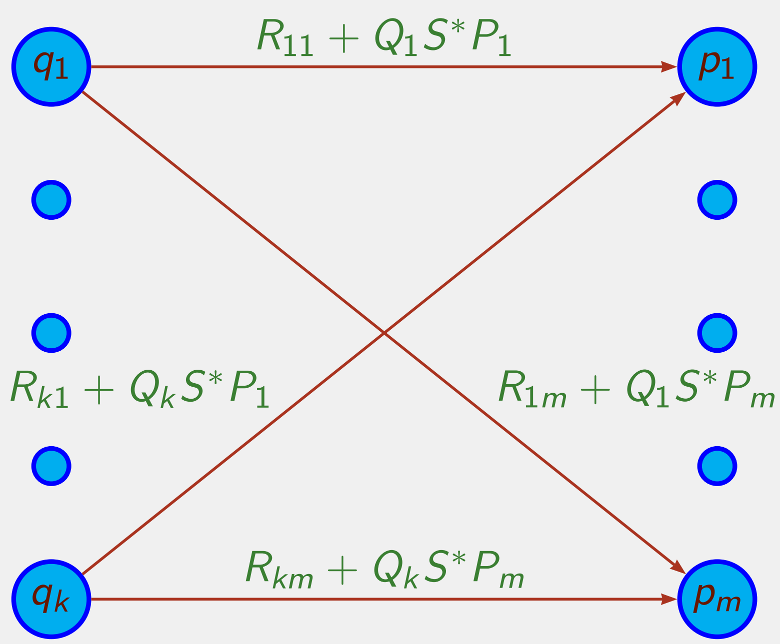
Let the FA A be:



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.

From FA to RE: Eliminating State s in A



From FA to RE: Eliminating State s in A

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.

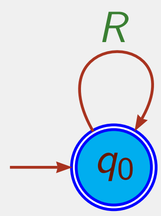
Labels now are not just symbols but (possible an infinite number of) strings: hence we will use RE as labels.

From FA to RE: Eliminating States in A

For *each accepting* state q we eliminate states until we have q_0 and q left.

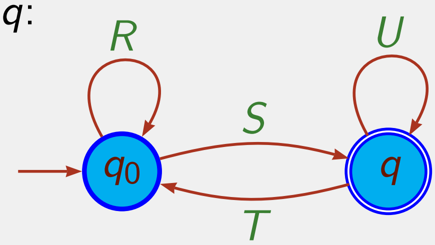
For each accepting state q we have 2 cases: $q_0 = q$ or $q_0 \neq q$.

If $q_0 = q$:



The expression is R^*.

If $q_0 \neq q$:

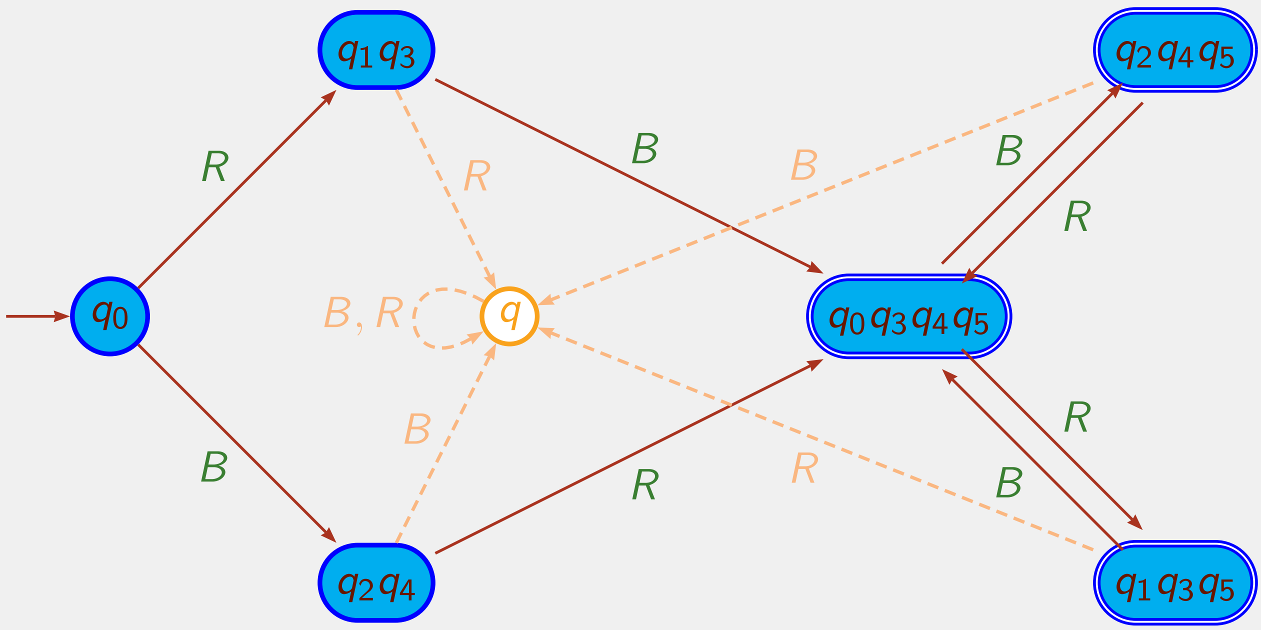


The expression is $(R + SU^*T)^*SU^*$.

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

Example: RE Representing Gilbreath's Principle

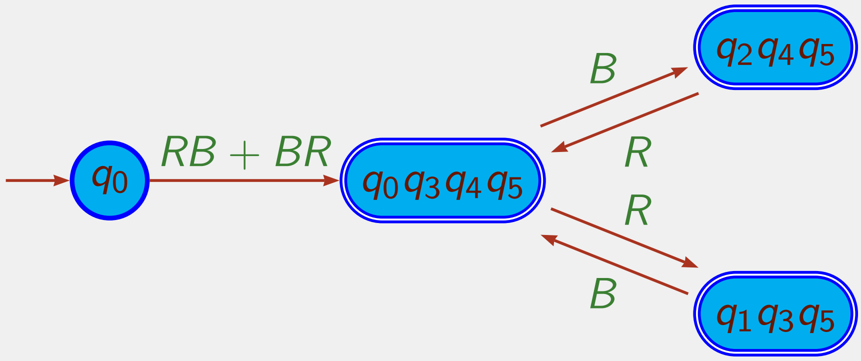
Recall:



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

Example: RE 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: RE Representing Gilbreath's Principle

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

Let us first do some simplifications.

$$\begin{aligned}(RB + BR)(RB)^*B(R(RB)^*B)^* &= (RB + BR)(RB)^*(BR(RB)^*)^*B && \text{by shifting}\\ &= (RB + BR)(RB + BR)^*B && \text{by the shifted-denesting rule}\\ &= (RB + BR)^+B\end{aligned}$$

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 FA we associate a system of equations with REs as solution.

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.

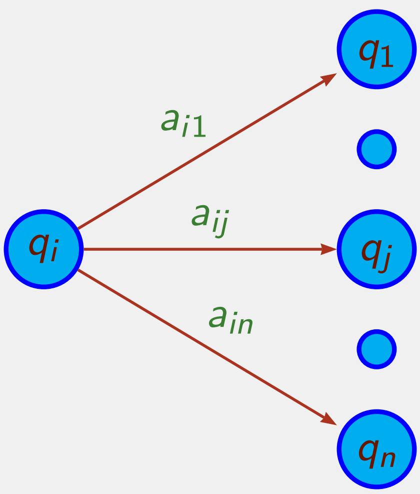
The solution to the linear system of equations associates a RE to each variable E_i.

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

From FA to RE: Constructing the Linear Equation System

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

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



Here 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

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

From FA to RE: 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: (sketch) 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).

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: RE Representing Gilbreath's Principle

We obtain the following system of equations (see slide 19):

$$
\begin{array}{ll}
E_0 = RE_{13} + BE_{24} & E_{0345} = \epsilon + BE_{245} + RE_{135} \\
E_{13} = BE_{0345} + RE_q & E_{245} = \epsilon + RE_{0345} + BE_q \\
E_{24} = RE_{0345} + BE_q & E_{135} = \epsilon + BE_{0345} + RE_q \\
 & E_q = (B + R)E_q
\end{array}
$$

Since $E_q = (B + R)^*\emptyset = \emptyset$, this can be simplified to:

$$
\begin{array}{ll}
E_0 = RE_{13} + BE_{24} & E_{0345} = \epsilon + BE_{245} + RE_{135} \\
E_{13} = BE_{0345} & E_{245} = \epsilon + RE_{0345} \\
E_{24} = RE_{0345} & E_{135} = \epsilon + BE_{0345}
\end{array}
$$

Example: RE Representing Gilbreath's Principle

And further to:

$$E_0 = (RB + BR)E_{0345}$$
$$E_{0345} = (RB + BR)E_{0345} + \epsilon + B + R$$

Then a solution to E_{0345} is

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

and the RE which is the solution to the problem is

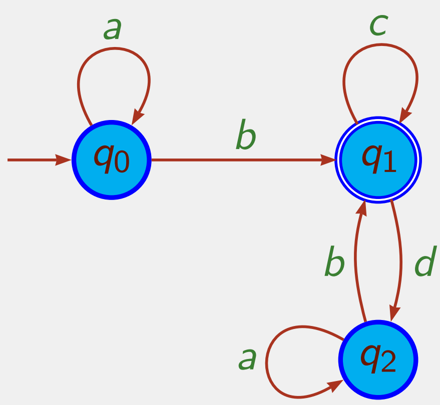
$$(RB + BR)(RB + BR)^*(\epsilon + B + R)$$

or

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

Example: Eliminating States

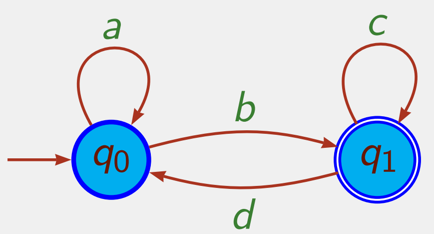
Consider the automaton D



By eliminating states the expression is

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

Consider the automaton D'



By eliminating states the expression is

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

But intuitively these automata are equivalent...

Example: Linear Equation System

The linear equations corresponding to the automaton D' are

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

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 be that $a^*b(c + da^*b)^* = (a + bc^*d)^*bc^*$! (see proof in slide 10.)

Exercise: What RE do we obtain for the automaton D?

Overview of Next Lecture

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.