Finite Automata and Formal Languages

TMV026/DIT321– LP4 2011

Lecture 7

April 5th 2011

Overview of today's lecture:

- **Equivalence between Finite Automata and Regular Expressions**
- **Pumping Lemma for Regular Languages**

From FA to RE: Eliminating States in an Automaton A

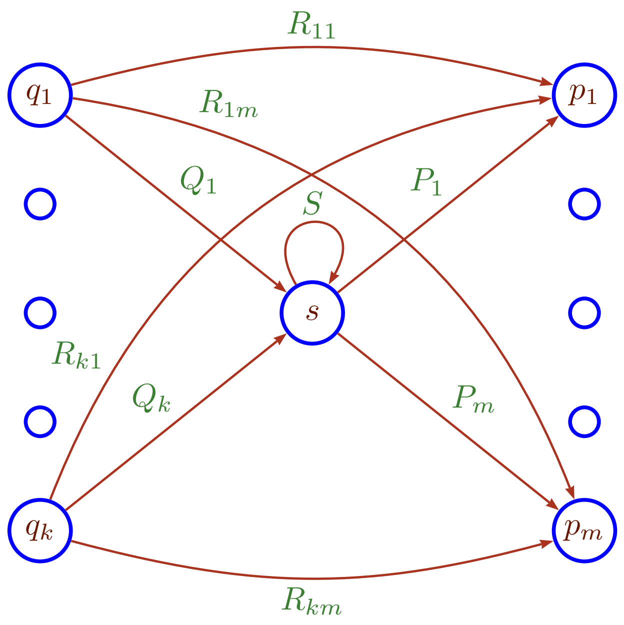
This method of constructing a RE from a FA involves eliminating states.

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.

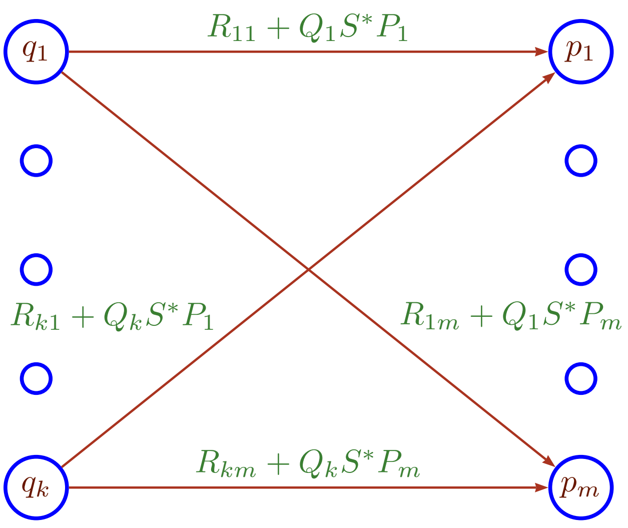
Eliminating State s in A



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.

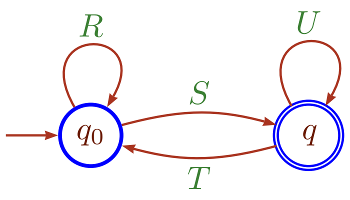
Eliminating State s in A



Eliminating States in A

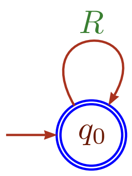
For *each accepting* state q we proceed as before until we have only q_0 and q left. For each q we have 2 cases: $q_0 \neq q$ or $q_0 = q$.

If $q_0 \neq q$:



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

If $q_0 = q$:

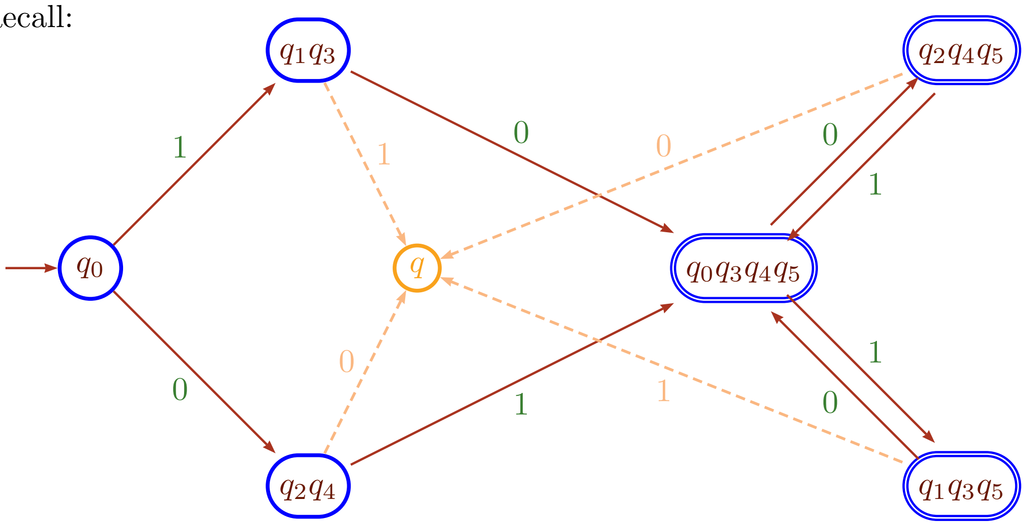


The expression is R^*

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

Example: Regular Expression Representing Gilbreath's Principle

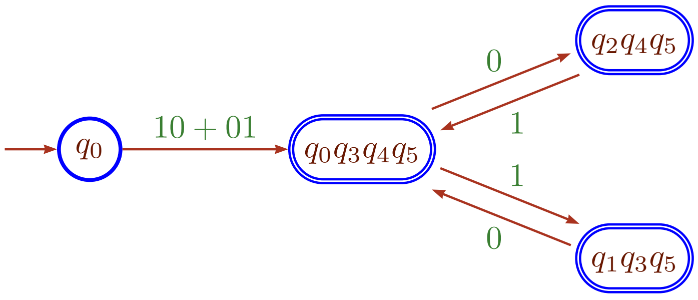
Recall:



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 $q_0q_3q_4q_5$: $(10+01)(10+01)^* = (10+01)^+$
- RE when final state is $q_2q_4q_5$: $(10+01)(10)^*0(1(10)^*0)^*$
- RE when final state is $q_1q_3q_5$: $(10+01)(01)^*1(0(01)^*1)^*$

Example: Regular Expression Representing Gilbreath's Principle

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

Let us first do some simplifications.

$$
\begin{aligned}
(10+01)(10)^*0(1(10)^*0)^* &= (10+01)(10)^*(01(10)^*)^*0 \qquad \text{by shifting}\\
&= (10+01)(10+01)^*0 \qquad \text{by the shifted-denesting rule}\\
&= (10+01)^+0
\end{aligned}
$$

Similarly $(10+01)(01)^*1(0(01)^*1)^* = (10+01)^+1$.

Hence the final RE is

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

which is equivalent to

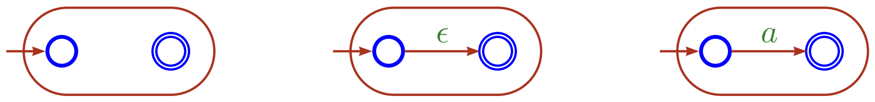
$$(10+01)^+(\epsilon+0+1)$$

From Regular Expressions to Finite Automata

Proposition: *Every language defined by a RE is accepted by a FA.*

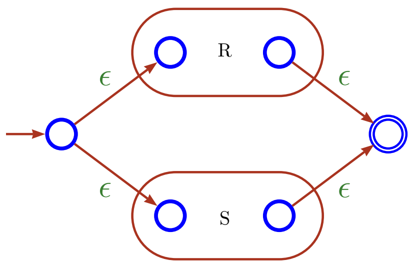
Proof: Let $\mathcal{L} = \mathcal{L}(R)$ for some RE R. By induction on R we construct a ϵ-NFA E with only one final state and no arcs into the initial state or out of the final state, and such that $\mathcal{L} = \mathcal{L}(E)$.

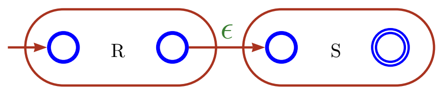
Basis cases are $\emptyset$, ϵ and $a \in \Sigma$. The corresponding ϵ-NFA recognising the languages $\emptyset$, $\{\epsilon\}$ and $\{a\}$ respectively, are:

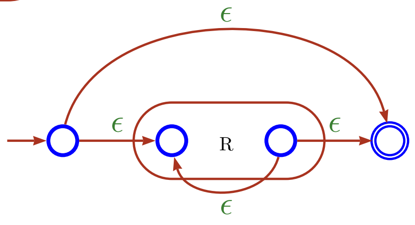


From RE to FA: Inductive Step

Given the RE R and S and FA for them, we construct the FA for $R + S$, RS and R^* recognising the languages $\mathcal{L}(R) \cup \mathcal{L}(S)$, $\mathcal{L}(R)\mathcal{L}(S)$ and $\mathcal{L}(R)^*$ respectively:

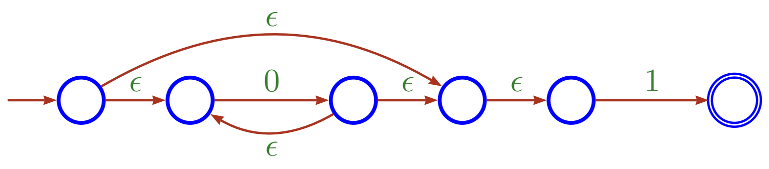




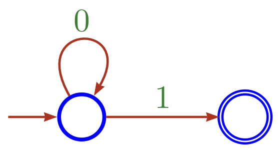


Example: From RE to FA

Let us follow this method to construct a FA for the RE 0^*1.



Compare it with the following FA:



How to Identify Regular Languages?

We have seen that a language is regular iff there is a DFA that accepts the language.

Then we saw that DFA, NFA and ϵ-NFA are equivalent in the sense that we can convert between them.

Hence FA accept all and only the regular languages (RL).

Now we have seen how to convert between FA and RE.

Thus RE also define all and only the RL.

How to Prove that a Language is NOT Regular?

In a FA with n states, any path

$$q_1 \stackrel{a_1}{\rightarrow} q_2 \stackrel{a_2}{\rightarrow} q_3 \stackrel{a_3}{\rightarrow} \ldots \stackrel{a_{m-1}}{\rightarrow} q_m \stackrel{a_m}{\rightarrow} q_{m+1}$$

has a loop if $m \geqslant n$.

That is, we have $i < j$ such that $q_i = q_j$ in the path above.

This can be seen as an application of the *Pigeonhole Principle*, which is an important reasoning technique in mathematics and computer science.

(See Wikipedia.)

The Pigeonhole Principle

"If you have more pigeons than pigeonholes and each pigeon flies into some pigeonhole, then there must be at least one hole with more than one pigeon."

More formally: if $f : X \rightarrow Y$ and $|X| > |Y|$ then f cannot be *injective* and there must exist at least 2 different elements with the same image, that is, there must exist $x, z \in X$ such that $x \neq z$ and $f(x) = f(z)$.

This principle is often used to show the existence of an object without building this object explicitly.

Example: In a room with at least 13 people, at least 2 of them are born the same month (maybe on different years).

We know the existence of these 2 people, maybe without being able to know exactly who they are.

How to Prove that a Language is Not Regular?

Example: Let us prove that $\mathcal{L} = \{0^m1^m | m \geqslant 0\}$ is not a RL.

Let us assume it is: then $\mathcal{L} = \mathcal{L}(A)$ for some FA A with n states.

Let $k \geqslant n > 0$ and let $w = 0^k1^k \in \mathcal{L}$.

Then there must be an accepting path $q_0 \xrightarrow{w} q \in F$.

Since there are only n states and $k \geqslant n$ we know there is a loop (by the pigeonhole principle) at some point when reading the 0's.

Then $w = xyz$ with $|xy| = j \leqslant n$, $y \neq \epsilon$ and $z = 0^{k-j}1^k$ such that

$$q_0 \xrightarrow{x} q_l \xrightarrow{y} q_l \xrightarrow{z} q \in F$$

Observe that the following path is also an accepting path

$$q_0 \xrightarrow{x} q_l \xrightarrow{z} q \in F$$

However y must be of the form 0^i with $i > 0$ hence $xz = 0^{k-i}1^k \notin \mathcal{L}$.

This contradicts the fact that A accepts $\mathcal{L}$.

The Pumping Lemma for Regular Languages

Theorem: *Let $\mathcal{L}$ be a RL. Then, there exists a constant n (which depends on $\mathcal{L}$) such that for every string $w \in \mathcal{L}$ and $|w| \geqslant n$, we can break w into 3 strings x, y and z such that $w = xyz$ and*

1. $y \neq \epsilon$
2. $|xy| \leqslant n$
3. $\forall k \geqslant 0, xy^kz \in \mathcal{L}$

Proof of the Pumping Lemma

Assume we have a FA A that accepts the language, then $\mathcal{L} = \mathcal{L}(A)$.

Let n be the number of states in A.

Then any path of length $m \geqslant n$ has a loop.

Let us consider $w = a_1 a_2 \ldots a_m \in \mathcal{L}$.

We have an accepting path and a loop such that

$$q_0 \xrightarrow{x} q_l \xrightarrow{y} q_l \xrightarrow{z} q \in F$$

with $w = xyz \in \mathcal{L}$, $y \neq \epsilon$, $|xy| \leqslant n$.

Then we also have

$$q_0 \xrightarrow{x} q_l \xrightarrow{y^k} q_l \xrightarrow{z} q \in F$$

for any k, that is, $\forall k \geqslant 0, xy^kz \in \mathcal{L}$.

Application of the Pumping Lemma

Example: Let us use the Pumping lemma to prove that $\{0^m1^m | m \geqslant 0\}$ is not a RL.

We assume it is.

Let n be the constant given by the lemma and let $w = 0^n1^n$, hence $|w| \geqslant n$.

By the lemma we know that $w = xyz$ with $y \neq \epsilon$, $|xy| \leqslant n$ and $\forall k \geqslant 0, xy^kz \in \mathcal{L}$.

Since $y \neq \epsilon$ and $|xy| \leqslant n$, we know that $y = 0^i$ with $i \geqslant 1$.

However, we have a contradiction since $xy^kz \notin \mathcal{L}$ for $k \neq 1$.

Note: The Pumping lemma is connected to the fact that a FA has *finite memory*! If we could build a machine with infinitely many states it would be able to recognise the language.

Another Application of the Pumping Lemma

Example: Let us prove that $\mathcal{L} = \{0^i 1^j | i \leqslant j\}$ is not a RL.

Let n be given by the Pumping lemma and let $w = 0^n 1^{n+1} \in \mathcal{L}$, hence $|w| \geqslant n$.

Then we know that $w = xyz$ with $y \neq \epsilon$, $|xy| \leqslant n$ and $\forall k \geqslant 0, xy^k z \in \mathcal{L}$.

Since $y \neq \epsilon$ and $|xy| \leqslant n$, we know that $y = 0^r$ with $r \geqslant 1$.

However, we have a contradiction since $xy^k z \notin \mathcal{L}$ for $k > 2$.

(Even for $k = 2$ if $r > 1$.)

Example: What about the languages $\{0^i 1^j \mid i \geqslant j\}$, $\{0^i 1^j \mid i > j\}$ and $\{0^i 1^j \mid i \neq j\}$? Does the Pumping lemma help?

Pumping Lemma is not a Necessary Condition

By showing that the Pumping lemma does not apply to a certain language $\mathcal{L}$ we prove that $\mathcal{L}$ is not regular.

However, if the Pumping lemma *does* apply to $\mathcal{L}$, I *cannot* conclude whether $\mathcal{L}$ is regular or not!

Example: We know $\mathcal{L} = \{b^m c^m \mid m \geqslant 0\}$ is not regular.

Let us consider $\mathcal{L}' = a^+\mathcal{L} \cup (b + c)^*$.

$\mathcal{L}'$ is not regular. If $\mathcal{L}'$ would be regular, then we can prove that $\mathcal{L}$ is regular (can be proved with the closure properties we will see soon).

However, the Pumping lemma does apply for $\mathcal{L}'$ with $n = 1$.

This shows the Pumping lemma is not a necessary condition for a language to be regular.