Finite Automata and Formal Languages

TMV026/DIT321– LP4 2012

Lecture 8
Ana Bove

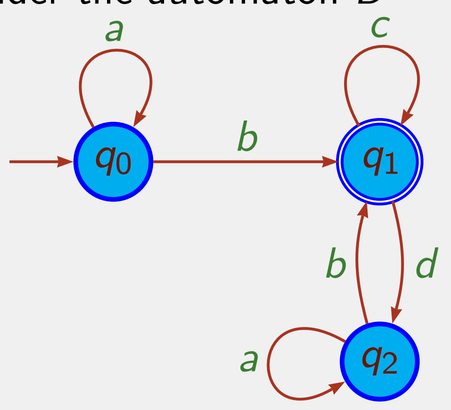
April 16th 2012

Overview of today's lecture:

- Small recap from FA to RE
- From RE to FA
- Pumping Lemma for Regular Languages
- Closure Properties for Regular Languages

Example: Eliminating States

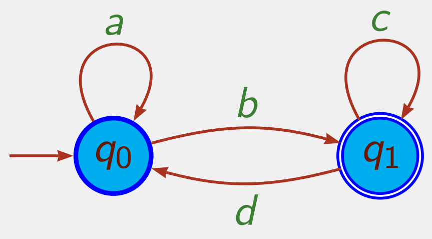
Consider the automaton D



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

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

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 10 lecture 7.)

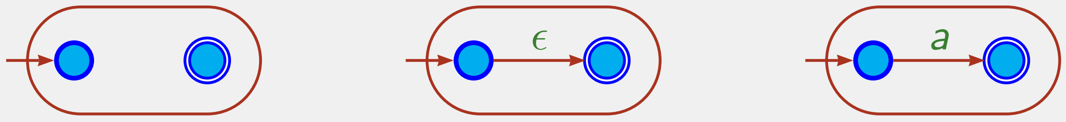
What RE do we obtain for the automaton D?

From Regular Expressions to Finite Automata

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

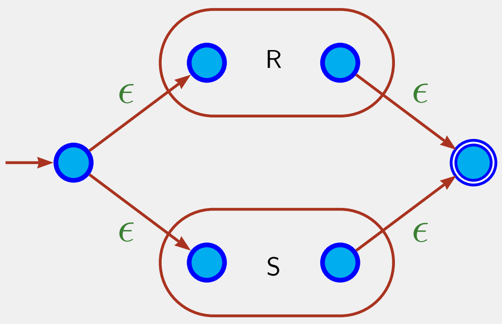
Proof: Let $\mathcal{L} = \mathcal{L}(R)$ for some RE R. By induction on R we construct a ϵ-NFA E with only one final state and no arcs into the initial state or out of the final state, and such that $\mathcal{L} = \mathcal{L}(E)$.

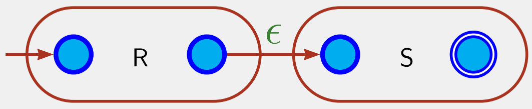
Base cases are $\emptyset$, ϵ and $a \in \Sigma$. The corresponding ϵ-NFA recognising the languages $\emptyset$, $\{\epsilon\}$ and $\{a\}$ respectively, are:

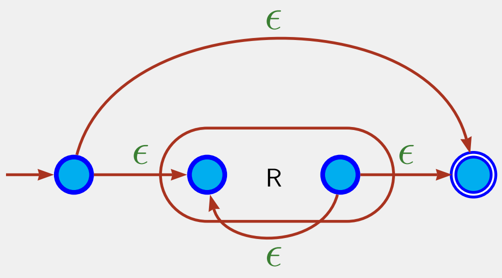


From RE to FA: Inductive Step

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:

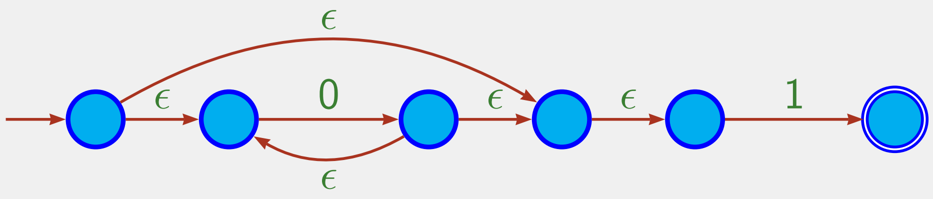




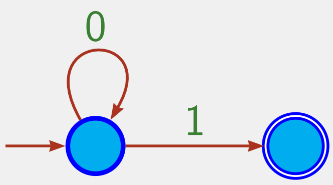


Example: From RE to FA

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

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

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 $f : X \to Y$ 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 $k \geqslant n$ we know there is a loop (by the pigeonhole principle) 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. $\forall k \geqslant 0, xy^k z \in \mathcal{L}$

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, $\forall k \geqslant 0, xy^k z \in \mathcal{L}$.

Example: Application of the Pumping Lemma

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.

Example: Application of the Pumping Lemma

Example: Let us prove that $\mathcal{L} = \{0^i1^j | i \leqslant j\}$ is not a RL.

Let n be given by the Pumping lemma and let $w = 0^n1^{n+1} \in \mathcal{L}$, hence $|w| \geqslant n$.

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

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

However, we have a contradiction since $xy^kz \notin \mathcal{L}$ for $k > 2$.
(Even for $k = 2$ if $r > 1$.)

Example: What about the languages $\{0^i1^j \mid i \geqslant j\}$, $\{0^i1^j \mid i > j\}$ and $\{0^i1^j \mid i \neq j\}$? 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 $\mathcal{L}$, we *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 (using the closure properties we will see next).
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.

Closure Properties for Regular Languages

Let $\mathcal{L}$ and $\mathcal{M}$ be RL. Then $\mathcal{L} = \mathcal{L}(R) = \mathcal{L}(D)$ and $\mathcal{M} = \mathcal{L}(S) = \mathcal{L}(F)$ for RE R and S, and DFA D and F.

We have seen that RL are closed under the following operations:

- Union : $\mathcal{L} \cup \mathcal{M} = \mathcal{L}(R + S)$ or $\mathcal{L} \cup \mathcal{M} = \mathcal{L}(D \oplus F)$ (slide 19, lect. 4);
- Complement : $\overline{\mathcal{L}} = \mathcal{L}(\overline{D})$ (slide 20, lect. 4);
- Intersection : $\mathcal{L} \cap \mathcal{M} = \overline{\overline{\mathcal{L}} \cup \overline{\mathcal{M}}}$ or $\mathcal{L} \cap \mathcal{M} = \mathcal{L}(D \times F)$ (sl. 18, lect. 4);
- Difference : $\mathcal{L} - \mathcal{M} = \mathcal{L} \cap \overline{\mathcal{M}}$;
- Concatenation : $\mathcal{L}\mathcal{M} = \mathcal{L}(RS)$;
- Closure (“star” operation) : $\mathcal{L}^* = \mathcal{L}(R^*)$;
- Prefix : Prefix($\mathcal{L}$) See exercise 2 on DFA.
  (Hint: in D, make final all states in a path from the start state to final state)

Closure under Prefix

Another way to prove that the language of prefixes of a RL is regular is as follows.

Define the following function over RE:

$$
\begin{aligned}
&pre(\emptyset) = \emptyset\\
&pre(\epsilon) = \epsilon\\
&pre(a) = \epsilon + a\\
&pre(R_1 + R_2) = pre(R_1) + pre(R_2)\\
&pre(R_1 R_2) = pre(R_1) + R_1 pre(R_2)\\
&pre(R^*) = R^* pre(R)
\end{aligned}
$$

and prove that $\mathcal{L}(pre(R)) = \text{Prefix}(\mathcal{L}(R))$.

Then, if $\mathcal{L} = \mathcal{L}(R)$ for some RE R then
$\text{Prefix}(\mathcal{L}) = \text{Prefix}(\mathcal{L}(R)) = \mathcal{L}(pre(R))$.