COMP4141 Theory of Computation

Lecture 4 Regular Languages cont.

Ron van der Meyden

CSE, UNSW

Revision: 2013/03/14

(Credits: David Dill, Thomas Wilke, Kai Engelhardt, Peter Höfner, Rob van Glabbeek)

Regular Expression Definition

Regular expressions are defined relative to some alphabet Σ.
This is a recursive definition of the structure of regular expressions. The structure of regular expressions is the basis for other recursive definitions and induction proofs. Every recursive definition and proof has to handle these six cases:

- a is a regular expression, if $a \in \Sigma$.
- $\emptyset$ is a regular expression.
- ϵ is a regular expression.
- $R_1 \cup R_2$ is a regular expression if R_1 and R_2 are.
- $R_1 \circ R_2$ is a regular expression if R_1 and R_2 are.
- R^* is a regular expression if R is.

Parentheses can be added in the obvious places to override precedence: * has the highest precedence, followed by $\circ$, and finally $\cup$ which has the lowest precedence, so $a \cup b \circ c^* = a \cup (b \circ (c^*))$. The first three definitions can be considered base cases while the last three are inductive.

Language of a Regular Expression

Let R be a regular expression. The language $L(R)$ of R is defined recursively on the structure of R.

- Case R is a for some $a \in \Sigma$: $L(a) = \{a\}$
- Case R is $\emptyset$: $L(\emptyset) = \emptyset$
- Case R is ϵ: $L(\epsilon) = \{\epsilon\}$
- Case R is $R_1 \cup R_2$: $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
- Case R is $R_1 \circ R_2$: $L(R_1 \circ R_2) = L(R_1)L(R_2)$
- Case R is R_1^*: $L(R_1^*) = (L(R_1))^*$

NB

L relates each syntactic object to a semantic object, whence we also call it a semantics for reg. exps.

$$L : RE_\Sigma \longrightarrow 2^{\Sigma^*}$$

From Regular Expressions to Finite Automata

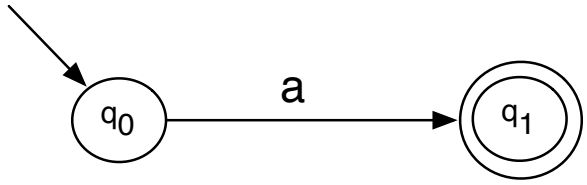
Theorem

For every regular expression R, there is an ϵ-NFA N_R such that $L(R) = L(N_R)$.

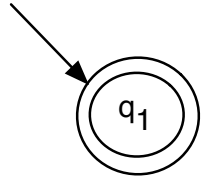
Proof.

by induction on the structure of R. For each of the three base cases and each of the three inductive cases we define an ϵ-NFA which recognises $L(R)$. □

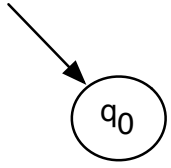
From Regular Expressions to Finite Automata cont.



a



ε



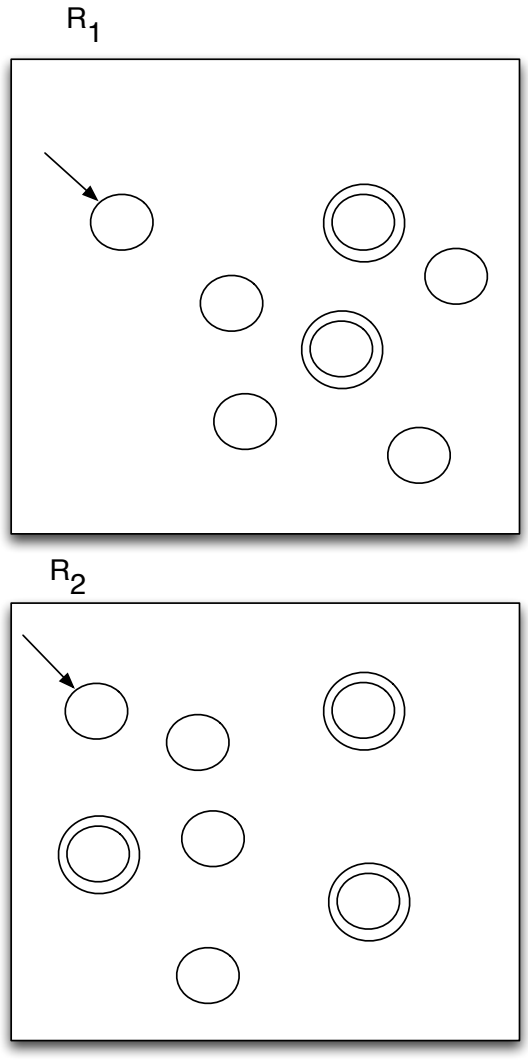
∅

From Regular Expressions to Finite Automata cont.

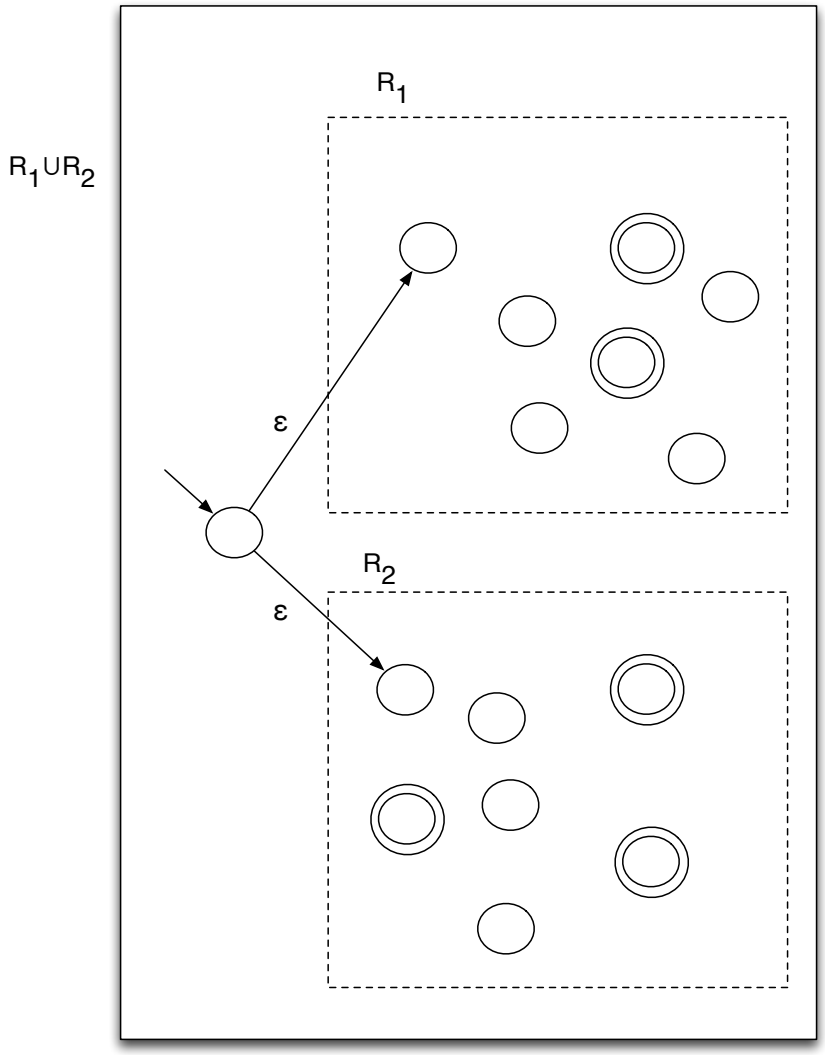
Base Cases (formally):

- $N_a = (\{q_0, q_1\}, \Sigma, \delta_a, q_0, \{q_1\})$ where $\delta_a(q_0, a) = \{q_1\}$ and $\delta_a(q, s) = \emptyset$, for all $(q, s) \neq (q_0, a)$.
- $N_\emptyset = (\{q_0\}, \Sigma, \delta_\emptyset, q_0, \emptyset)$ where $\delta_\emptyset(q_0, s) = \emptyset$ for all $s \in \Sigma \cup \{\epsilon\}$.
- $N_\epsilon = (\{q_0\}, \Sigma, \delta_\emptyset, q_0, \{q_0\})$.

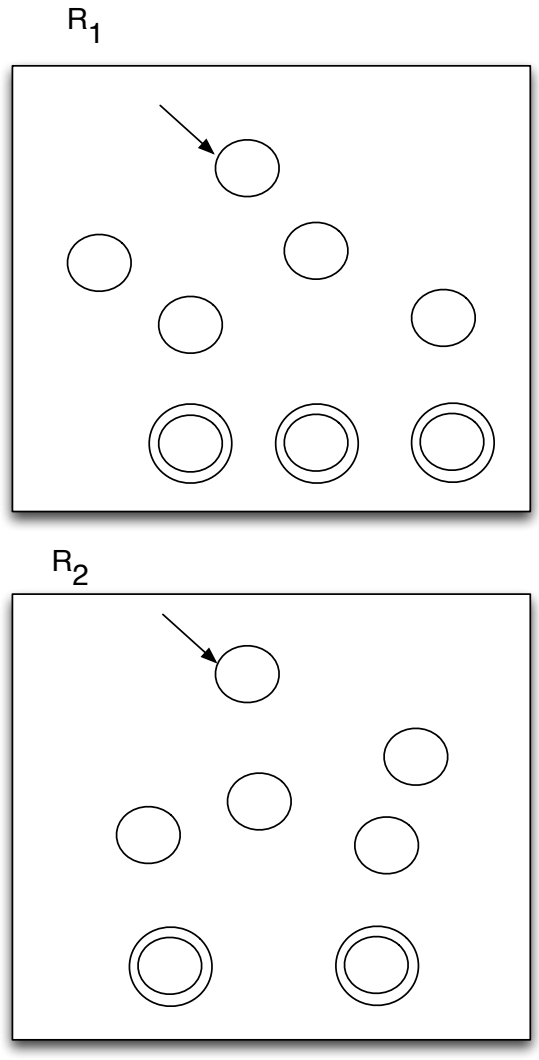
From Regular Expressions to Finite Automata (Union)



From Regular Expressions to Finite Automata (Union)

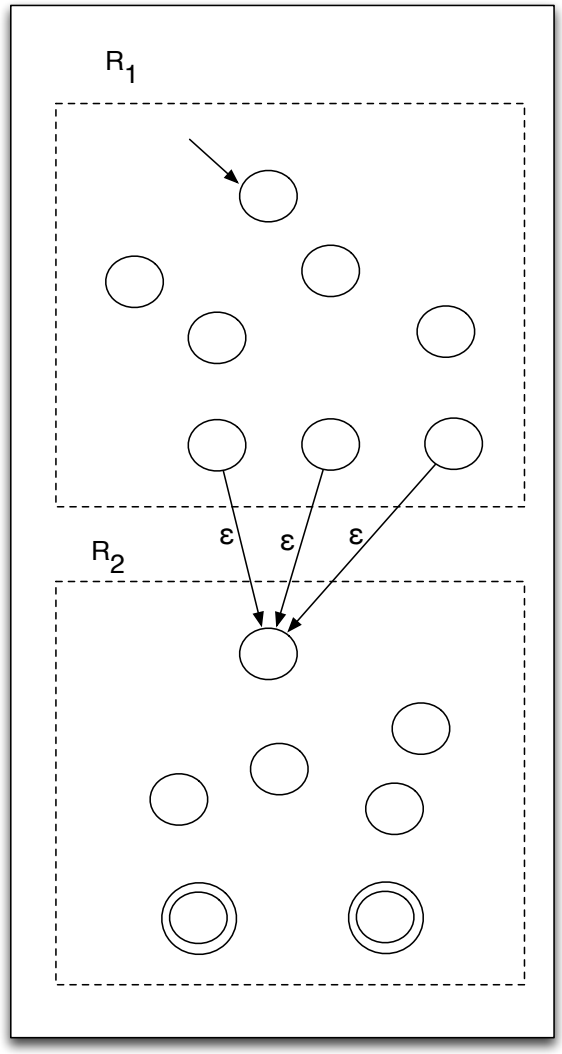


From Regular Expressions to Finite Automata (Concatenation)

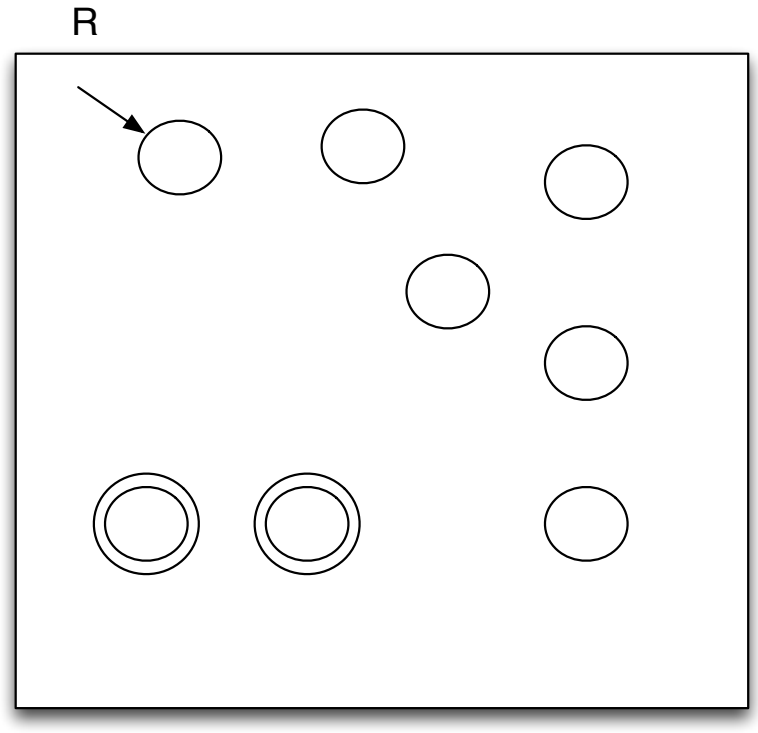


From Regular Expressions to Finite Automata (Concatenation)

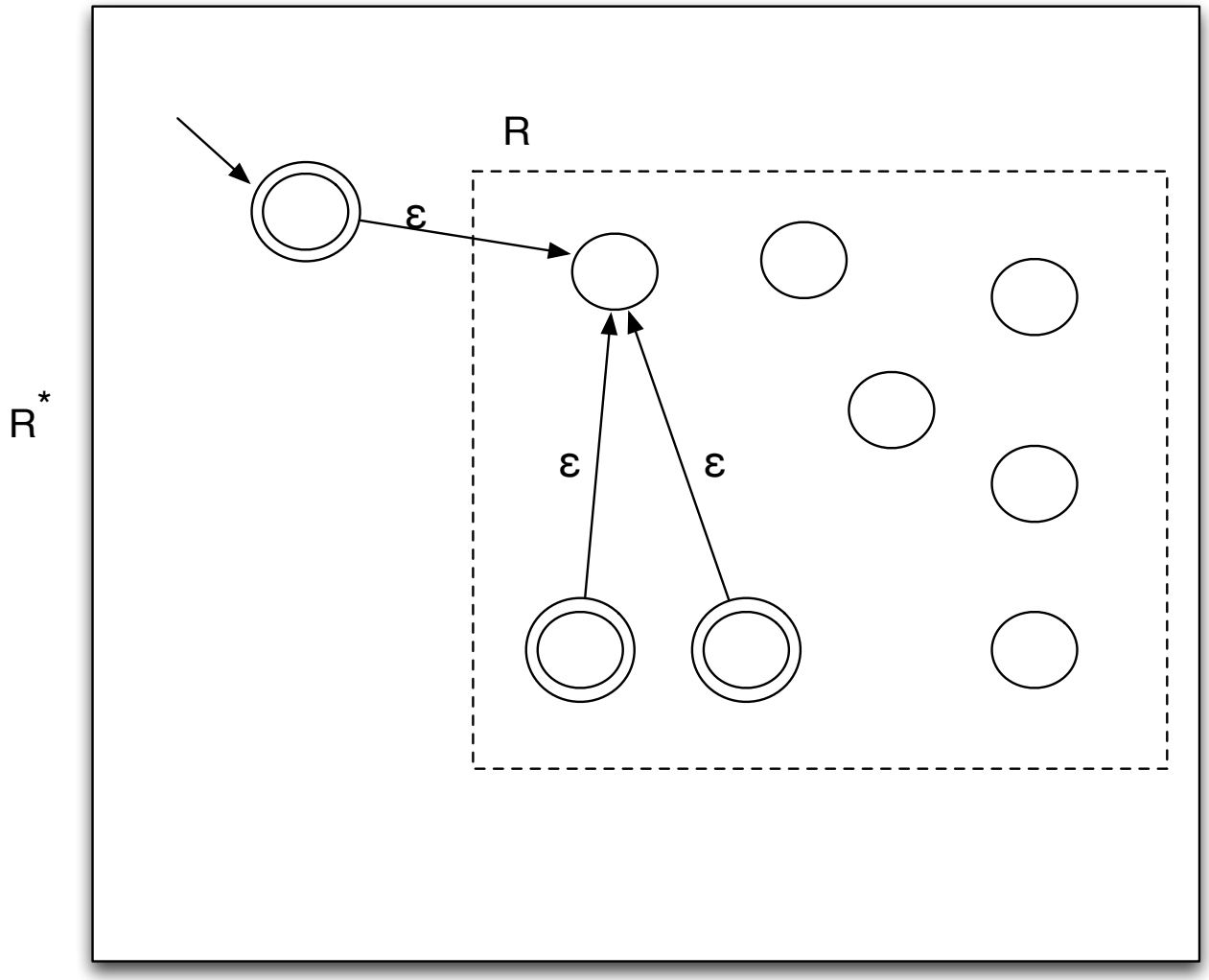
R_1R_2



From Regular Expressions to Finite Automata (Kleene Star)



From Regular Expressions to Finite Automata (Kleene Star)



From Regular Expressions to Finite Automata cont.

Inductive Cases (formally):

- For $i \in \{1,2\}$ let $N_i = (Q_i, \Sigma, \delta_i, q_0^{(i)}, F_i)$ be an ϵ-NFA satisfying $L(N_i) = L(R_i)$ according to the induction hypothesis. Moreover we assume w.l.o.g. that the two sets of states, Q_1 and Q_2 are disjoint and do not contain q_0.
  $N_{R_1 \cup R_2} = (\{q_0\} \cup Q_1 \cup Q_2, \Sigma, \delta_1 \cup \delta_2 \cup$
  $\left\{ (q_0, \epsilon) \mapsto q_0^{(i)} \mid 1 \leq i \leq 2 \right\}, q_0, F_1 \cup F_2)$.
- With the same N_i as in the previous case, define
  $N_{R_1 \circ R_2} = (Q_1 \cup Q_2, \Sigma, \delta_1 \cup \delta_2 \cup$
  $\left\{ (q, \epsilon) \mapsto \delta_1(q, \epsilon) \cup \{q_0^{(2)}\} \mid q \in F_1 \right\}, q_0^{(1)}, F_2)$.
- Let $N = (Q, \Sigma, \delta, q_0', F)$ be an ϵ-NFA satisfying $L(N) = L(R_1)$ according to the induction hypothesis. W.l.o.g. assume that $q_0 \notin Q$. Define
  $N_{R_1^*} = (\{q_0\} \cup Q, \Sigma, \delta \cup \{ (q, \epsilon) \mapsto q_0' \mid q \in F \cup \{q_0\} \}, q_0, F)$.

Implications

We already know that ϵ-NFAs, NFAs, and DFAs have equal expressive power.

The latest theorem shows that FA are at least as expressive as regular expressions.

Next: Can we do anything with a FA that we cannot do with a regular expression?

From FA to Regular Expressions

Theorem

For every finite automaton (any type) A, there is a regular expression R such that $L(R) = L(A)$.

FA $\rightarrow$ RE

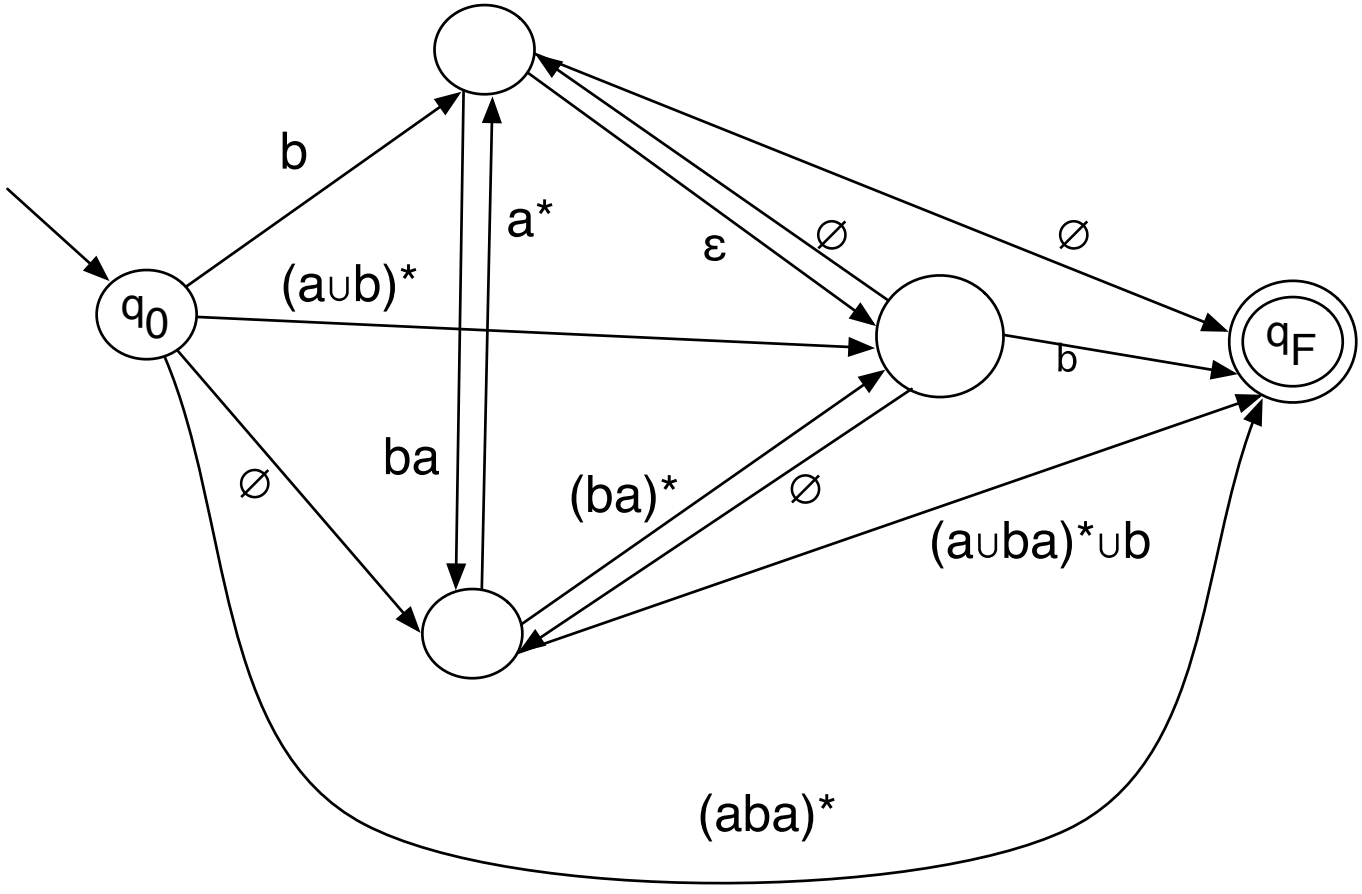
Let Σ be an alphabet. We write RE_Σ for the set of all regular expressions over Σ.

Roadmap: DFA $\longrightarrow$ GNFA $\longrightarrow$ RE_Σ

where a GNFA is an NFA with

- regular expressions instead of symbols as labels on transitions,
- a unique final state,
- and a full transition relation, except that there are no transitions either (a) into the start state or (b) out of the accept state.

GNFA example



GNFAs

Definition

A *generalised non-deterministic finite automaton (GNFA)* is a 5-tuple $(Q, \Sigma, \delta, q_0, q_F)$ where

1. Q is finite set of states,
2. Σ is the input alphabet,
3. $\delta : (Q \setminus \{q_F\}) \times (Q \setminus \{q_0\}) \longrightarrow \mathrm{RE}_\Sigma$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $q_F \in Q$ is the accept state.

Language of a GNFA

Definition

A GNFA accepts a string $w \in \Sigma^*$ if there exists a $k \in \mathbb{N}$, a sequence of states $q_1, \ldots, q_k$, and a sequence of strings $w_1, \ldots, w_k$ such that

1. $w = w_1 \ldots w_k$
2. $w_i \in L(\delta(q_{i-1}, q_i))$, for $i \in \{1, \ldots, k\}$
3. $q_k = q_F$

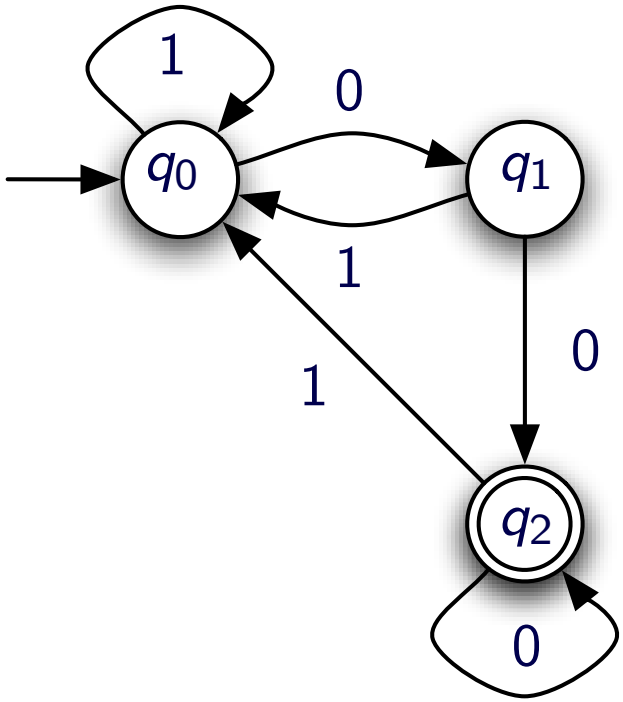
Note that this already implies that execution started at q_0.

DFA $\longrightarrow$ GNFA

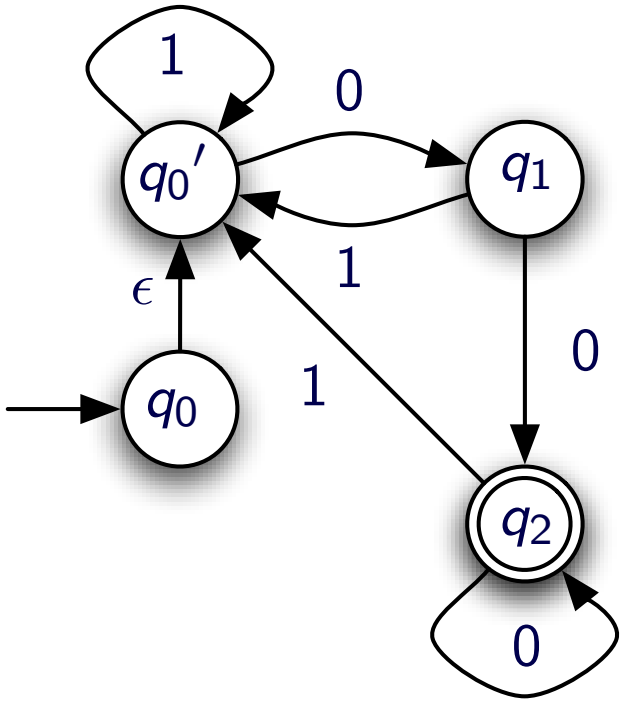
This is straightforward.

- Add a new start state, connect it via an ϵ transition to the original start state (which is no longer the initial state).
- Add a new accept state, connect all the original accept states via ϵ transition to it. (They are no longer accept states.)
- Between pairs (q, q') of original states, replace all existing transitions by a single one labeled with the (regular expression) union of the labels on the original transitions from q to q'.
- Introduce $\emptyset$-labeled transitions where needed (e.g. from the new start state to all but the old start state).

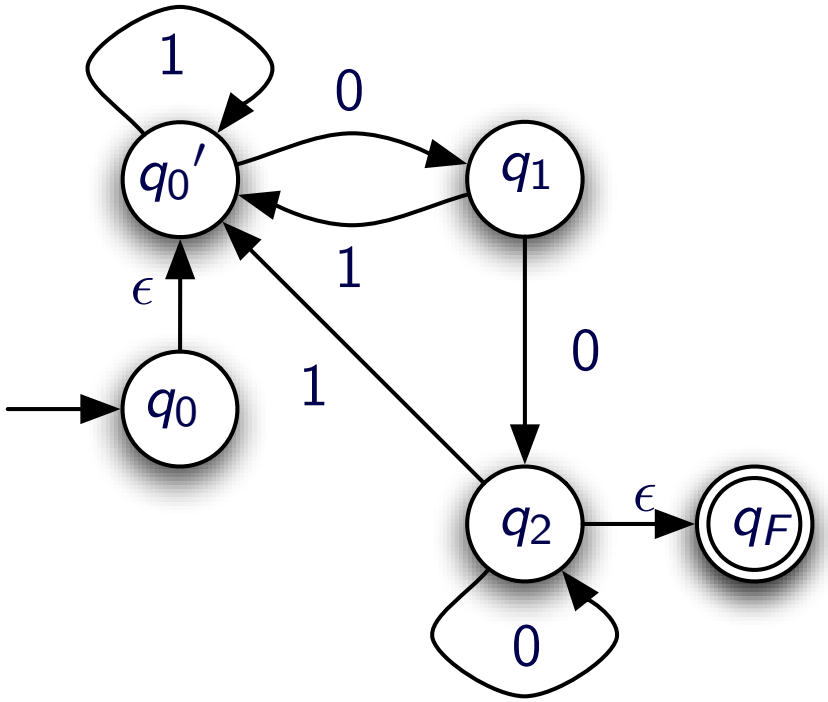
Example DFA $\longrightarrow$ GNFA



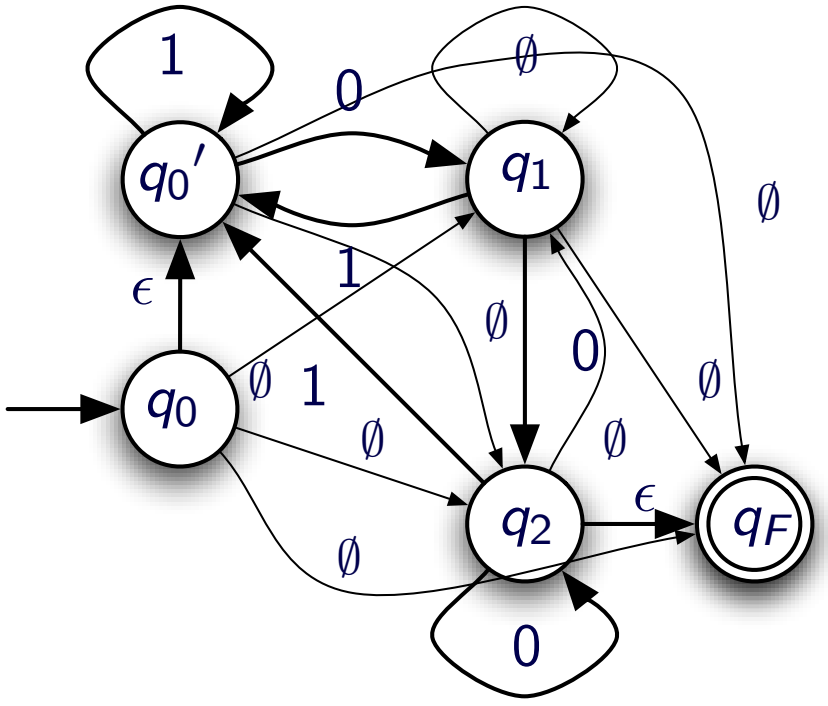
Example DFA $\longrightarrow$ GNFA



Example DFA $\longrightarrow$ GNFA



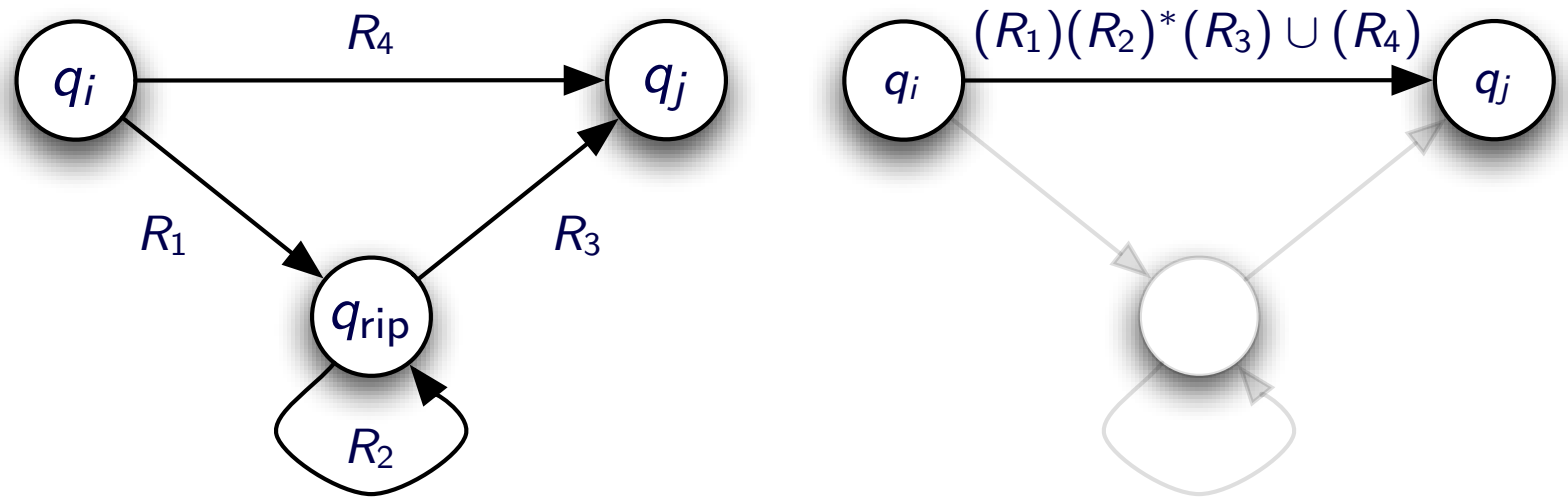
Example DFA $\longrightarrow$ GNFA



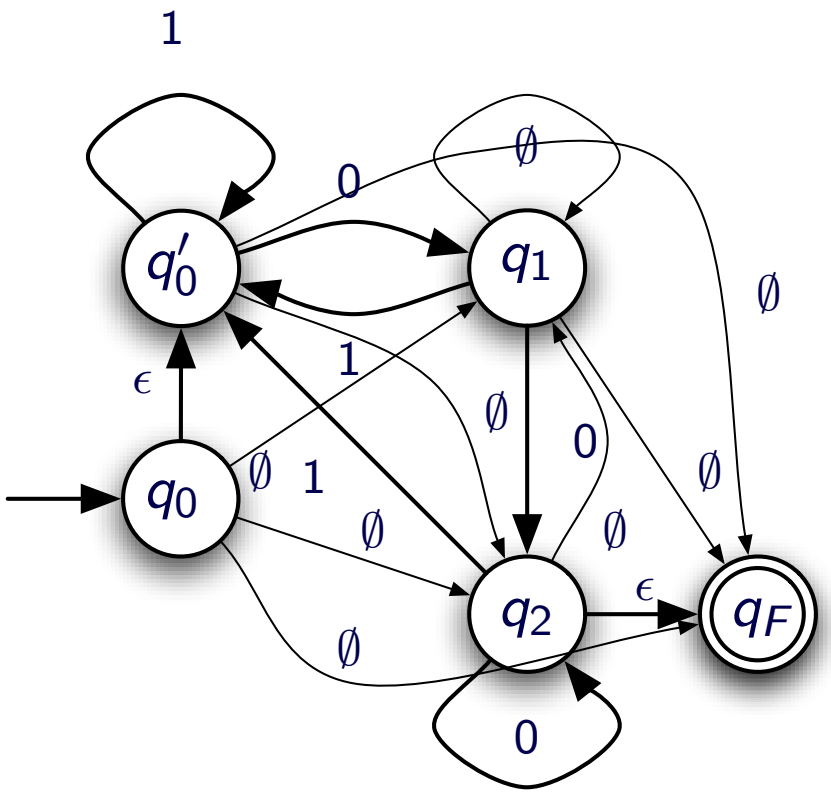
GNFA $\longrightarrow$ $\mathbf{RE}_\Sigma$

The gist of this construction is to eliminate all *inner* states of the GNFA (i.e. the states of the original DFA) one by one. When all of them are gone, only q_0 and q_F remain. The only remaining transition (from q_0 to q_f) is labeled with the regular expression we're after.

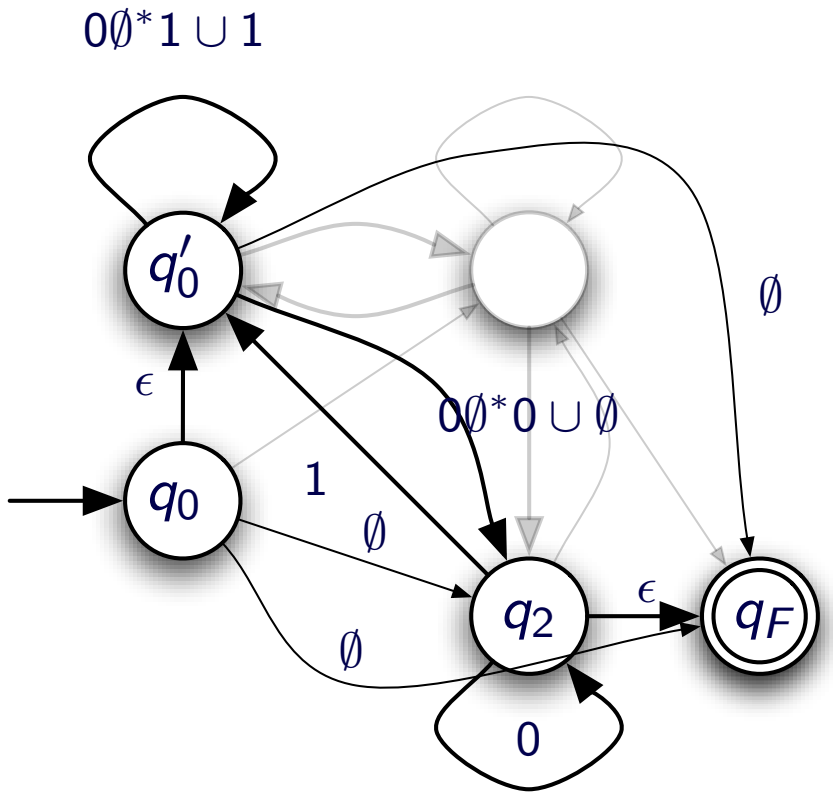
To eliminate an "inner" state q_{rip}, we need to augment the labels on every transition from $q_i \neq q_{\text{rip}}$ to $q_j \neq q_{\text{rip}}$:



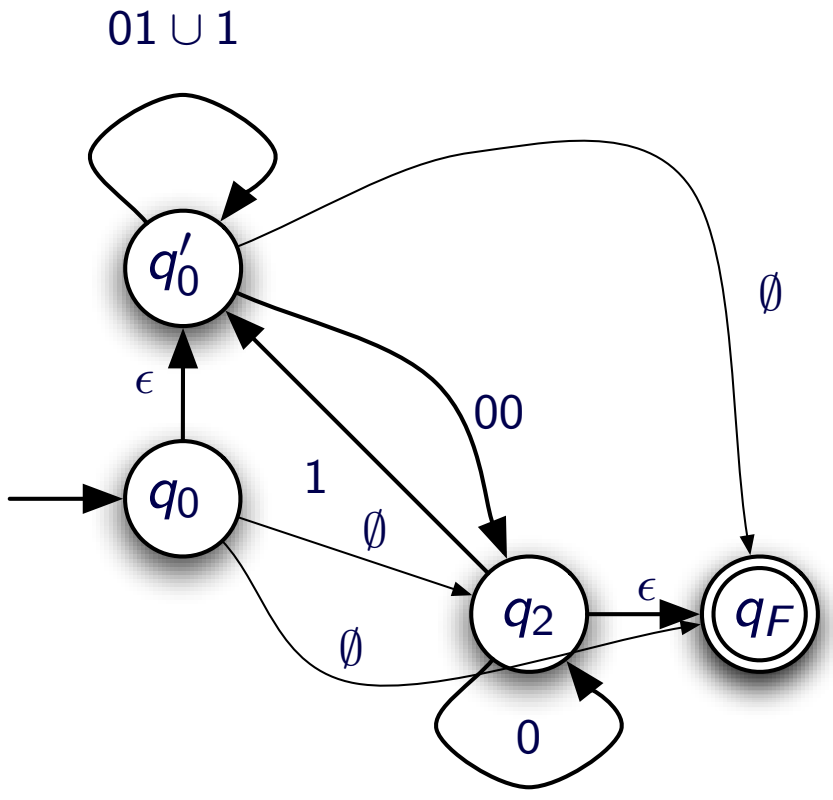
Example GNFA $\longrightarrow$ $\mathbf{RE}_\Sigma$



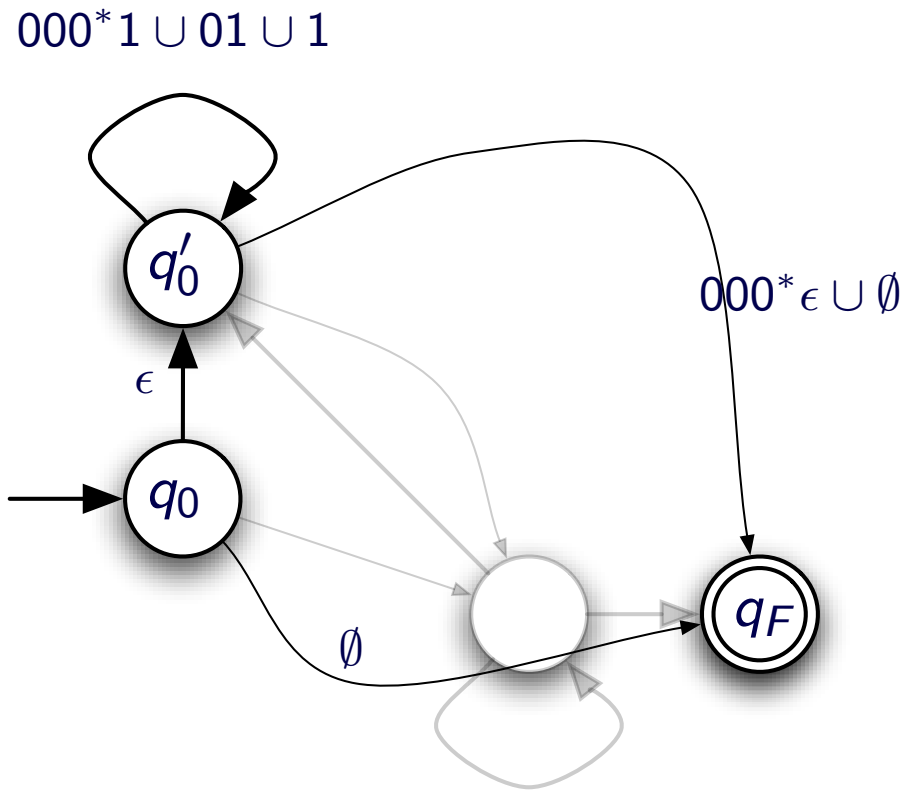
Example GNFA $\longrightarrow$ RE_Σ



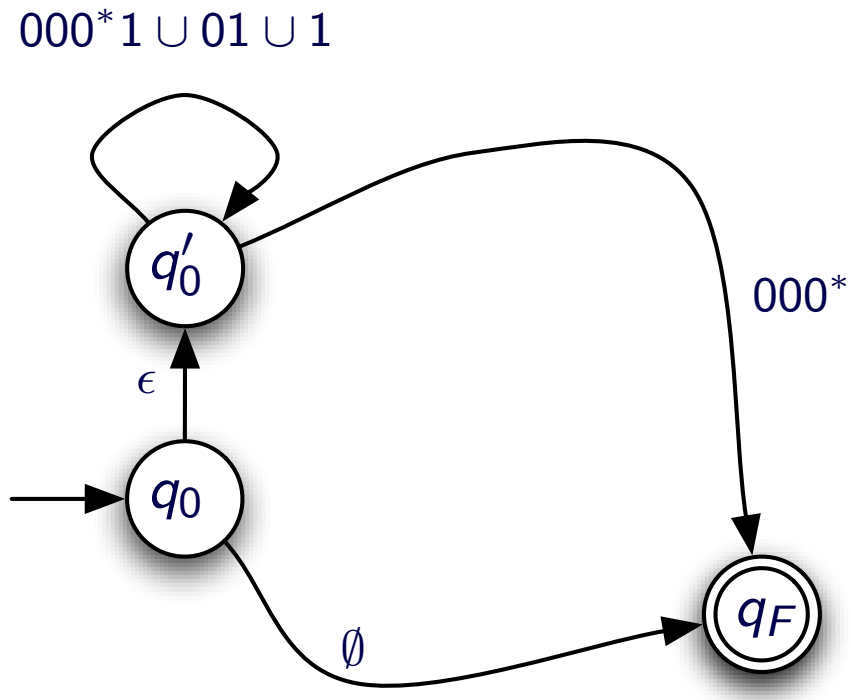
Example GNFA $\longrightarrow$ $\mathbf{RE}_\Sigma$



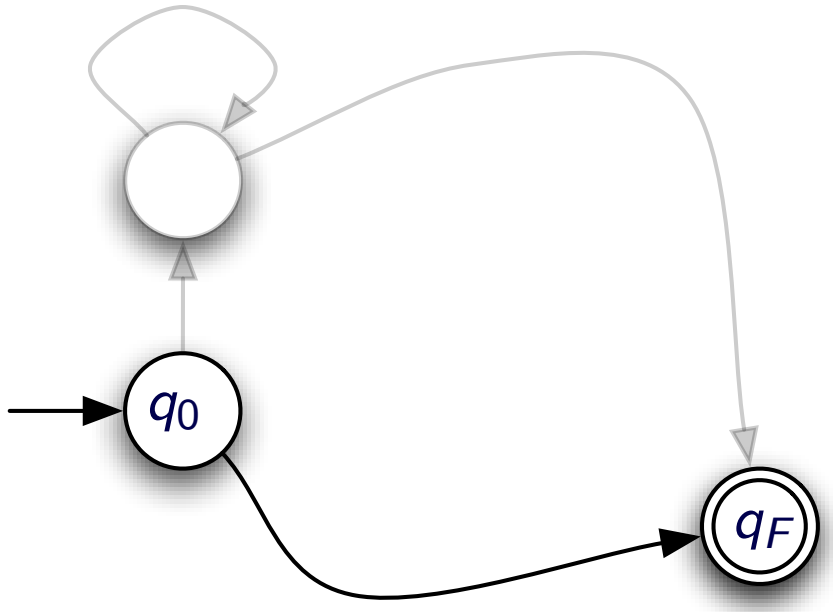
Example GNFA $\longrightarrow$ RE_Σ



Example GNFA $\longrightarrow$ RE_Σ

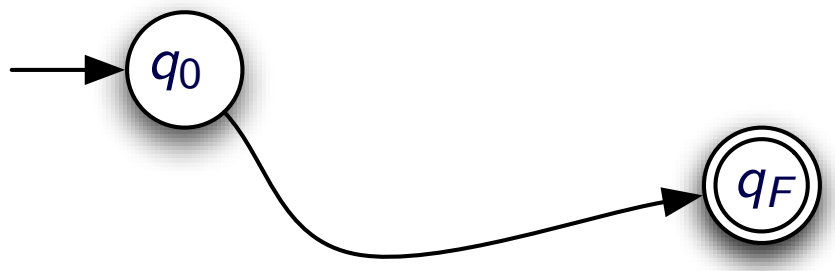


Example GNFA $\longrightarrow$ RE_Σ



$\epsilon(000^*1 \cup 01 \cup 1)^*000^* \cup \emptyset$

Example GNFA $\longrightarrow$ $\mathbf{RE}_\Sigma$



$(000^*1 \cup 01 \cup 1)^*000^*$

How to Prove Non-Regularity

Finite automata have their limits when it comes to characterising languages. A DFA with k states must have visited at least one of its states repeatedly to accept a word of length $> k$. It has gone through a loop. The DFA must therefore also accept all words generated by going through that loop any number of times.
This is what we call *pumping*.

Theorem (Pumping Lemma)

If $L \subseteq \Sigma^*$ is regular then there exists $p \in \mathbb{N}$ (the pumping length) where, if $w \in L$ with $|w| \geq p$, then w may be split into three pieces, $w = xyz$, satisfying the following conditions:

1. $xy^i z \in L$, *for all* $i \in \mathbb{N}$,
2. $|y| > 0$, *and*
3. $|xy| \leq p$.

Proof of the Pumping Lemma

Proof.

Let L be any regular language. We know there is a DFA A that accepts the same language. Let n be the number of states in that DFA. Let w be any string in L such that $|w| \geq n$. Let $u = a_1 a_2 \ldots a_n$ be the prefix of w such that $|u| = n$. Consider the sequence of states $p_0 p_1 \ldots p_n$ such that $p_0 = q_0$ and $\hat{\delta}(p_0, a_1 a_2 \ldots a_i) = p_i$. There are $n + 1$ positions in the sequence of states, so, by the pigeonhole principle, there is at least one state that appears at two distinct positions; call the first position i and the second j . So, $0 \leq i < j \leq n$ and $p_i = p_j$. So w can be broken up into xyz where $\hat{\delta}(q_0, x) = p_i$, $\hat{\delta}(p_i, y) = p_j = p_i$ and $\hat{\delta}(p_j, z) \in F$. $|xy| \leq n$ by the pigeonhole principle. $y \neq \epsilon$ because $j > i$. Every string of the form $xy^k z$ for $k \geq 0$ is in L, because $\hat{\delta}(q_0, xy^k) = p_i$, hence $\hat{\delta}(q_0, xy^k z) = \hat{\delta}(p_i, z) \in F$. □

Non-Regularity Proofs with the Pumping Lemma

How can we use the Pumping Lemma to prove that a given language L is not regular? We use its contrapositive, that is:

$$L \text{ can't be pumped} \quad \Rightarrow \quad L \text{ is not regular.}$$

Non-Regularity Proof Example

Example

Consider $L_1 = \{ a^i b^i \mid i \in \mathbb{N} \}$. Assume to the contrary that L_1 is regular and that p is its pumping length. Choose $w = a^p b^p$. Next we show that regardless of how we split w into xyz, none of these splits satisfies all three conditions given in the Pumping Lemma.

Case y consists only of a's: Then $xyyz$ contains more as than bs and is thus not in L_1, violating condition 1.

Case y contains b's: Then $|xy| > p$, violating condition 3.

Case $y = \epsilon$: violates 2.

Limits of the Pumping Lemma

Consider the language $L_2 = \{ c^i a^j b^k \mid i = 1 \Rightarrow j = k + 1 \}$.

Is it regular?

No. Assume to the contrary that it is regular. We'll try to "chop off" the leading c to relate L_2 to L_1. Let's define the *left quotient* of a language B by a word w:

$$w \backslash B = \{ v \in \Sigma^* \mid wv \in B \}$$

Exercise: show $w \backslash B$ is regular when B is.

It follows that $ca \backslash L_2 = \{ a^i b^i \mid i \in \mathbb{N} \} = L_1$ would be regular, too. But we've just proved it is not regular. Contradiction!

Limits of the Pumping Lemma cont.

Assume again that L_2 is regular and that $p > 1$ is its pumping length. Can we lead this to a contradiction?

Let $w \in L_2$ such that $|w| \geq p$. We choose x and y (and implicitly z) such that $w = xyz$ depending on the number $\|w\|_c$ of cs in w.

Case $\|w\|_c = 0$: Choose $x = \epsilon$ and $y =$ first letter of w.

Case $0 < \|w\|_c \leq 3$: Choose $x = \epsilon$ and $y = c^{\|w\|_c}$.

Case $\|w\|_c > 3$: Choose $x = \epsilon$ and $y = cc$.

In each case we can pump without leaving L_2, that is, $xy^iz \in L_2$, for all $i \in \mathbb{N}$.

We conclude that

$$L \text{ can be pumped } \begin{matrix} \not\Rightarrow \\ \Leftarrow \end{matrix} L \text{ is regular}$$

The Myhill-Nerode Theorem

This theorem provides another exact characterisation of the regular languages. We use it to prove non-regularity—mostly when the pumping lemma fails.

Let $L \subseteq \Sigma^*$. If there exists a $z \in \Sigma^*$ such that $xz \in L$ and $yz \notin L$ (or vice versa), we call x and y *distinguishable by* L.

We write $x \equiv_L y$ if x and y are not distinguishable by L.

(Equivalently, $x \equiv_L y$ when $xz \in L \Leftrightarrow yz \in L$ for all $z \in \Sigma^*$.)

NB

$\equiv_L$ *is an equivalence relation on* Σ^*. ▸ what?

We write $[w]_L$ for the $\equiv_L$-equivalence class $\{ v \in \Sigma^* \mid w \equiv_L v \}$ of w.

Definition

The *index of L* is the number of its $\equiv_L$-equivalence classes.

The Myhill-Nerode Theorem

Theorem (Myhill-Nerode)

$L \subseteq \Sigma^*$ is regular iff the index of L is finite.

Sketch of proof.

"$\Rightarrow$": Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA with $L(A) = L$. Show that $|Q| \geq$ index of L.

"$\Leftarrow$": Assume the index of L is finite. Define the DFA

$$A_L = (\{\ [w]_L \mid\ w \in \Sigma^*\ \}, \Sigma, \delta_L, [\epsilon]_L, F_L)$$

where $\delta_L([w]_L, a) = [wa]_L$ and $F_L = \{\ [w]_L \mid\ w \in L\ \}$. Show that $L(A_L) = L$.

▸ details

□

Using the Myhill-Nerode Theorem

How can we use Myhill-Nerode to prove that a given language L is not regular? We use the contrapositive of the "$\Rightarrow$" direction of the theorem.

More specifically, it suffices to find an infinite sequence $u_0, u_1, \ldots$ of strings such that for all $i \neq j$ there exists a string w_{ij} such that $u_i w_{ij} \in L$ but $u_j w_{ij} \notin L$ (or vice versa).

It follows that $u_i \notin [u_j]_L$ for all distinct $i, j \in \mathbb{N}$ and hence there are infinitely many $\equiv_L$-equivalence classes, that is, the index of L is not finite.

Examples

- $L_1 = \{ a^i b^i \mid i \in \mathbb{N} \}$. Choose $u_i = a^i$ and $w_{ij} = b^i$.
- Recall $L_2 = \{ c^i a^j b^k \mid i = 1 \Rightarrow j = k + 1 \}$. Choose $u_i = ca^{i+1}$ and $w_{ij} = b^i$.

Reminder: Binary Relations

Let A and B be sets. Any $R \subseteq A \times B$ is called a *binary relation*. We often write aRb for $(a, b) \in R$.

If the first and second set are the same, i.e. $R \subseteq A^2$, we call R a *binary relation on* A and define the following properties of such relations. Say that R is:

- *reflexive* if aRa for all $a \in A$.
- *irreflexive* if aRa for no $a \in A$.
- *transitive* if $aRb \wedge bRc \Rightarrow aRc$ for all $a, b, c \in A$.
- *symmetric* if $aRb \Rightarrow bRa$ for all $a, b \in A$.
- *anti-symmetric* if $aRb \wedge bRa \Rightarrow a = b$ for all $a, b \in A$.
- an *equivalence relation* if it's reflexive, transitive, and symmetric.
- a *partial order* if it's reflexive, transitive, and anti-symmetric.
- a *strict partial order* if it's irreflexive, transitive, and anti-symmetric.

back

Proof of $A_L = L$

To prove that $A_L = L$ we show for all $w \in \Sigma^*$ that

$$\hat{\delta}_L([\epsilon]_L, w) = [w]_L \ . \tag{1}$$

Then we conclude that

$$\begin{aligned} w \in L(A_L) &\Leftrightarrow \hat{\delta}_L([\epsilon]_L, w) \in F_L && \text{def } L(\text{DFA}) \\ &\Leftrightarrow [w]_L \in F_L && \text{by } (1) \\ &\Leftrightarrow w \in L \end{aligned}$$

Proof of $\hat{\delta}_L([\epsilon]_L, w) = [w]_L$

Proof.

by induction on the length of w.

Base case: $\hat{\delta}_L([\epsilon]_L, \epsilon) = [\epsilon]_L$ by definition of $\hat{\delta}$ for DFAs.

Inductive case:

$$\begin{aligned}
\hat{\delta}_L([\epsilon]_L, va) &= \delta_L(\hat{\delta}_L([\epsilon]_L, v), a) && \text{def. } \hat{\delta} \text{ for DFAs} \\
&= \delta_L([v]_L, a) && \text{ind. hyp.} \\
&= [va]_L && \text{def. } \delta_L \quad \square
\end{aligned}$$

back