Formal Languages, Automata and Computation

DFAs to Regular Expressions

DFA Minimization – Closure Properties

Carnegie Mellon University in Qatar

SUMMARY

- Regular Expression (RE) define regular sets
- RE $\Rightarrow$ NFA $\Rightarrow$ DFA

Equivalence of REs to Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Lemma – The *only if* part

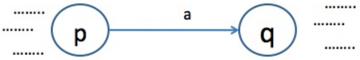
If a language is regular then it is described by a regular expression

Proof Idea

- Generalized transitions: label transitions with regular expressions
- Generalized NFAs (GNFA)
- Iteratively eliminate states of the GNFA one by one, until only two states and a single generalized transition is left.

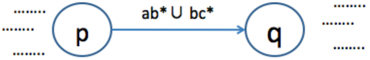
GENERALIZED TRANSITIONS

- DFAs have single symbols as transition labels



  - If you are in state p and the next input symbol matches a, go to state q

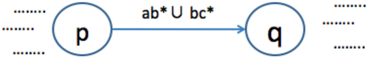
- Now consider



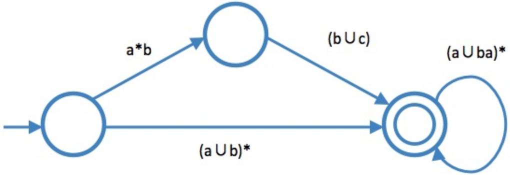
  - If you are in state p and a prefix of the remaining input matches the regular expression $\mathbf{ab}^* \cup \mathbf{bc}^*$ then go to state q

GENERALIZED TRANSITIONS AND NFA

- A generalized transition is a transition whose label is a regular expression

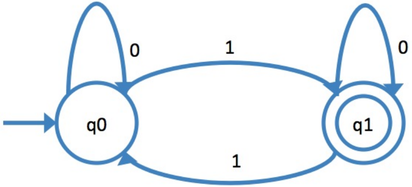


- A Generalized NFA is an NFA with generalized transitions.

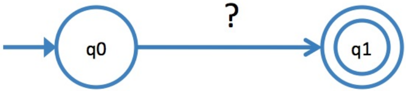


  - In fact, all standard DFA transitions are generalized transitions with regular expressions of a single symbol!

- Consider the 2-state DFA



$\downarrow$



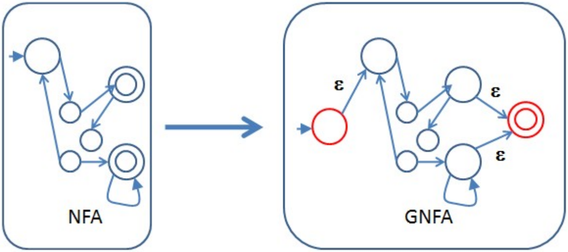
- $\mathbf{0}^*\mathbf{1}$ takes the DFA from state q_0 to q_1
- $(\mathbf{0} \cup \mathbf{10}^*\mathbf{1})^*$ takes the machine from q_1 back to q_1
- So $\mathbf{?} = \mathbf{0}^*\mathbf{1}(\mathbf{0} \cup \mathbf{10}^*\mathbf{1})^*$ represents all strings that take the DFA from state q_0 to q_1

GENERALIZED NFAS

- Take any DFA and transform it into a GNFA
  - with only two states: one start and one accept
  - with one generalized transition
- then we can “read” the regular expression from the label of the generalized transition (as in the example above)

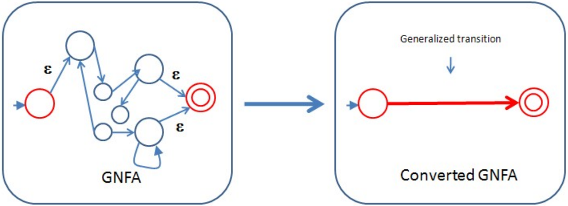
DFA TO GNFA

- We will add two new states to a DFA:
  - A new start state with an ϵ-transition to the original start state, but with no transitions from any other state
  - A new final state with an ϵ-transition from all the original final states, but with no transitions to any other state
- The previous start and final states are no longer!



Reducing a GNFA

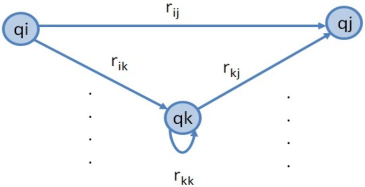
- We eliminate all states of the GNFA one-by-one leaving only the start state and the final state.



- When the GNFA is fully converted, the label of the only generalized transition is the regular expression for the language accepted by the original DFA.

ELIMINATING STATES

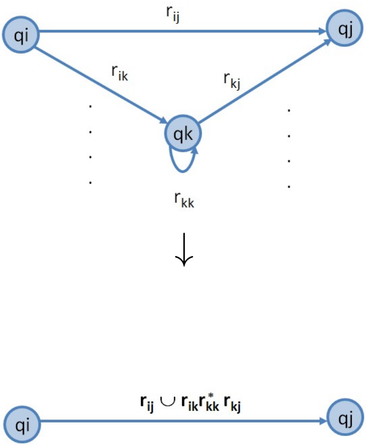
- Suppose we want to eliminate state q_k, and q_i and q_j are two of the remaining states ($i = j$ is possible).



- How can we modify the transition label between q_i and q_j to reflect the fact that q_k will no longer be there?
  - There are two paths between q_i and q_j
    - Direct path with regular expression $\mathbf{r_{ij}}$
    - Path via q_k with the regular expression $\mathbf{r_{ik} r_{kk}^{*} r_{kj}}$

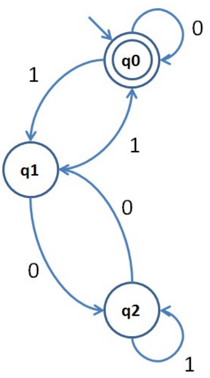
Eliminating States

- There are two paths between q_i and q_j
  - Direct path with regular expression $\mathbf{r_{ij}}$
  - Path via q_k with the regular expression $\mathbf{r_{ik} r_{kk}^* r_{kj}}$
- After removing q_k, the new label would be $\mathbf{r'_{ij}} = \mathbf{r_{ij}} \cup \mathbf{r_{ik} r_{kk}^* r_{kj}}$

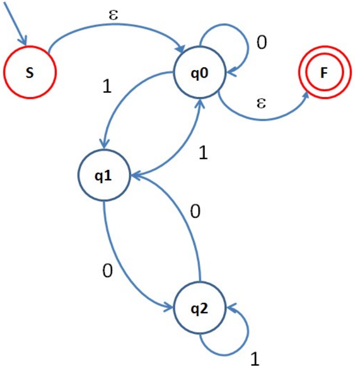


DFA-TO-GNFA-RE CONVERSION EXAMPLE

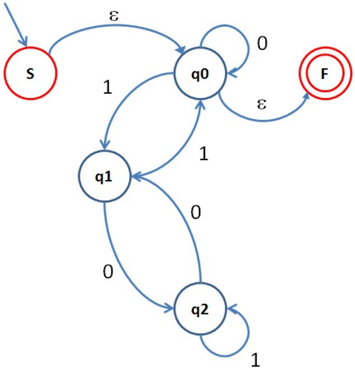
- DFA for binary numbers divisible by 3

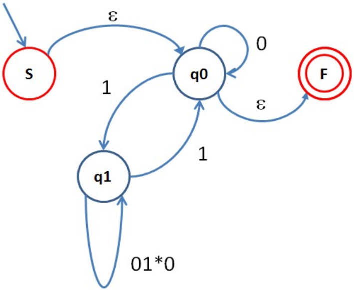


- Initial GNFA



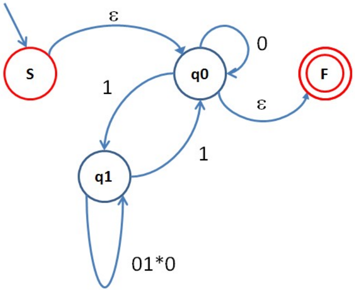
- Let's eliminate q_2

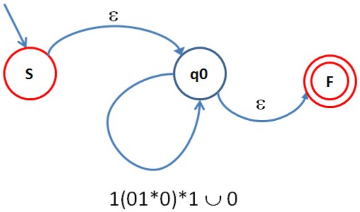




$$q_i = q_1, q_j = q_1, q_k = q_2$$

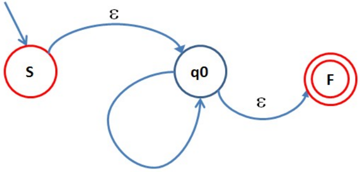
- Let's eliminate q_1

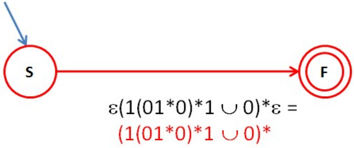




DFA-TO-GNFA-RE CONVERSION EXAMPLE

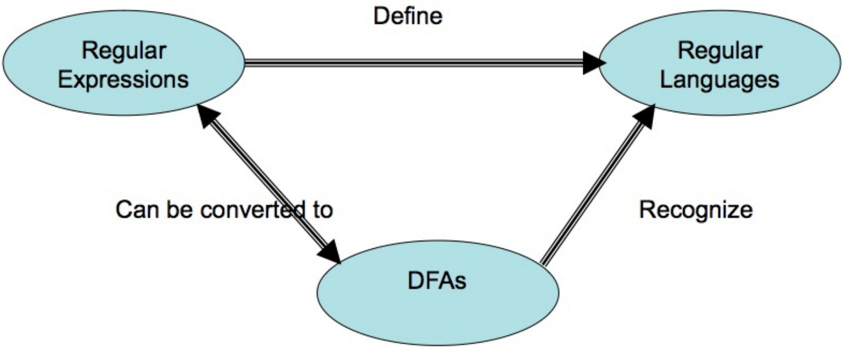
- Let's eliminate q_0



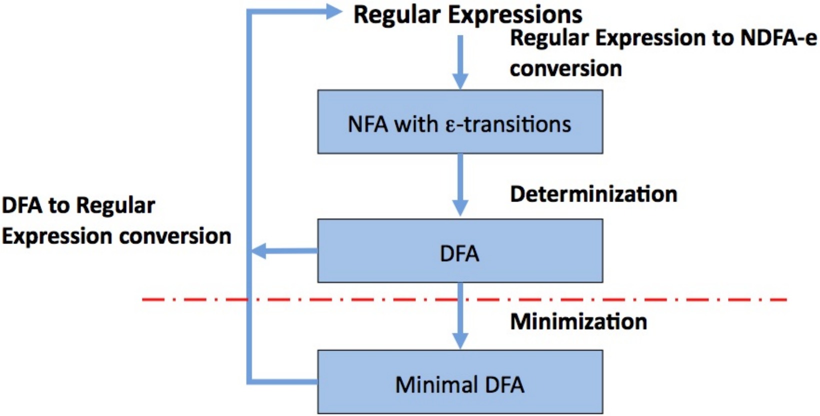


- So the regular expression we are looking for is $(\mathbf{1}(\mathbf{01}^*\mathbf{0})^*\mathbf{1} \cup \mathbf{0})^*$

The Story So Far

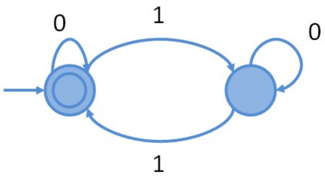


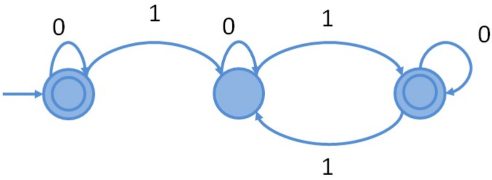
THE STORY SO FAR



DFA Minimization

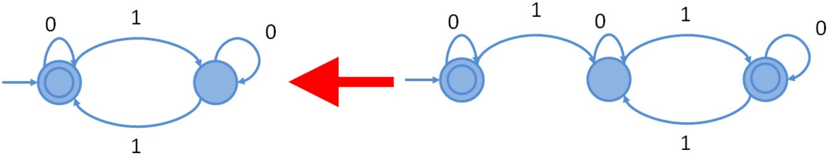
- Every DFA defines a unique language
- But in general, there may be many DFAs for a given language.
- These DFAs accept the same language.





DFA Minimization

- In practice, we are interested in the DFA with the minimal number of states
  - Use less memory
  - Use less hardware (flip-flops)



INDISGUISHABLE STATES

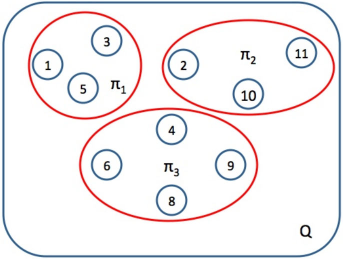
- Two states p and q of a DFA are called indistinguishable if for all $\omega \in \Sigma^*$,
  - $\delta^*(p, \omega) \in F \Leftrightarrow \delta^*(q, \omega) \in F$, and
  - $\delta^*(p, \omega) \notin F \Leftrightarrow \delta^*(q, \omega) \notin F$,
- Basically, these two states behave the same for all possible strings!
- Hence, a state p is distinguishable from state q
  - If there is at least one string ω such that either $\delta^*(p, \omega) \in F$ or $\delta^*(q, \omega) \in F$ and the other is not

INDISTINGUISHABILITY

- Indistinguishable states behave the same for all possible strings!
- So why have indistinguishable states? All but one can be eliminated!
- Indistinguishability is an equivalence relation
  - Reflexive: Each state is indistinguishable from itself
  - Symmetric: If p is indistinguishable from q, then q is indistinguishable from p
  - Transitive: If p is indistinguishable from q, and q is indistinguishable from r, then p is indistinguishable from r.

Indistinguishability and Partitions

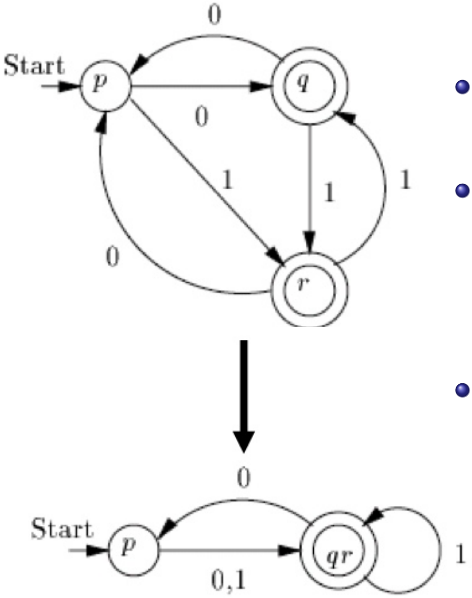
- Indistinguishability is an equivalence relation
  - Reflexive:Each state is indistinguishable from itself
  - Symmetric: If p is indistinguishable from q, then q is indistinguishable from p
  - Transitive: If p is indistinguishable from q, and q is indistinguishable from r, then p is indistinguishable from r.
- An equivalence relation on a set Q induces a partitioning $\pi = \{\pi_1, \pi_2, \cdots, \pi_k\}$ such that
  - For all i and j, $\pi_i \cap \pi_j = \Phi$,
  - $\bigcup_i \pi_i = Q$



Identifying Distinguishable States

- Basis: Any nonaccepting state is distinguishable from any accepting state ($\omega = \epsilon$).
- Induction: States *p* and *q* are distinguishable if there is some input symbol *a* such that $\delta(p, a)$ is distinguishable from $\delta(q, a)$.
- All other pairs of states are indistinguishable, and can be merged appropriately

IDENTIFYING DISTINGUISHABLE STATES

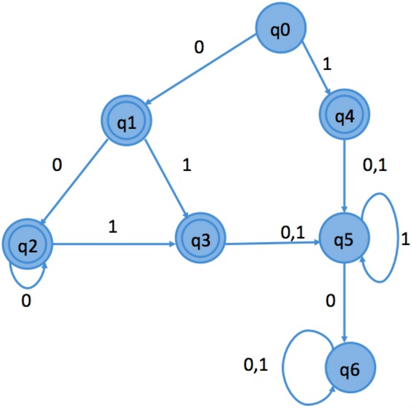


- p is distinguishable from q and r by basis
- Both q and r go to p with 0, so no string beginning with 0 will distinguish them
- Starting in either q and r, an input of 1 takes us to either, so they are indistinguishable.

The Procedure MARK

1. Remove all inaccessible states
2. Consider all pairs of states (p, q)
   - if $p \in F$ and $q \notin F$ or $p \notin F$ and $q \in F$, mark (p, q) as distinguishable
3. Repeat the following until no previously unmarked pairs are marked
   - $\forall p, q \in Q$ and $\forall a \in \Sigma$, find $\delta(p, a) = p'$ and $\delta(q, a) = q'$,
   - if (p', q') is marked distinguishable then mark (p, q) distinguishable.

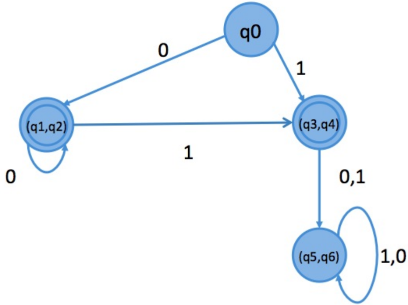
Minimization Example



- q_1 and q_2 are equivalent
- q_3 and q_4 are equivalent
- q_5 and q_6 are equivalent

	q_0	q_1	q_2	q_3	q_4	q_5
q_1	×					
q_2	×	✓				
q_3	×	×	×			
q_4	×	×	×	✓		
q_5	×	×	×	×	×	
q_6	×	×	×	×	×	✓

The Minimized DFA



- q_1 and q_2 are equivalent
- q_3 and q_4 are equivalent
- q_5 and q_6 are equivalent

q_1	×	■	■	■	■	■
q_2	×	✓	■	■	■	■
q_3	×	×	×	■	■	■
q_4	×	×	×	✓	■	■
q_5	×	×	×	×	×	■
q_6	×	×	×	×	×	✓
■	q_0	q_1	q_2	q_3	q_4	q_5

Is the minimized DFA really minimal?

- Let M be the DFA found by the previous procedure (with states $P = \{p_0, p_1, \ldots, p_m\}$)
- Suppose there is an equivalent DFA M_1 with δ_1 but with fewer states ($Q = \{q_0, q_1 \ldots, q_n\} n < m$).
- Since all states of M are distinguishable, there must be distinct strings, $\omega_1, \omega_2, \ldots, \omega_m$ such that $\delta^*(p_0, \omega_i) = p_i$ for all i.

IS THE MINIMIZED DFA REALLY MINIMAL?

- Since M_1 has fewer states than M, then there must be strings ω_k and ω_l (among the previous ω_i's) such that $\delta_1^*(q_0, \omega_k) = \delta_1^*(q_0, \omega_l)$ (Pigeonhole principle-see later)
- Since p_k and p_l are distinguishable, there must be some string x such that
  - $\delta^*(p_0, \omega_k \cdot x) = \delta^*(p_k, x)$ is a final state and $\delta^*(p_0, \omega_l \cdot x) = \delta^*(p_l, x)$ is NOT a final state, or vice versa. So $\omega_k \cdot x$ is accepted and $\omega_l \cdot x$ is not (or vice versa)

Is the minimized DFA really minimal?

- But

$$\begin{aligned}\delta_1^*(q_0, \omega_k \cdot x) &= \delta_1^*(\delta_1^*(q_0, \omega_k), x) \\ &= \delta_1^*(\delta_1^*(q_0, \omega_l), x) \\ &= \delta_1^*(q_0, \omega_l \cdot x)\end{aligned}$$

- So M_1 either accepts both $\omega_k \cdot x$ and $\omega_l \cdot x$ or rejects both. So M_1 and M can not be equivalent.
- So M_1 can not exist.

More on DFA Minimization

- DFA minimization is not covered in the textbook.
- See
  - `http://www.cs.uiuc.edu/class/sp06/cs273/Lectures/2005-slides/lec09.pdf`
  - Introduction to Automata Theory, Languages and Computation, by Hopcroft, Motwani and Ullman, Addison Wesley, 3^{rd} edition, Section 4.4

  for more formal details.

Closure Properties of Regular Languages

- Regular languages are closed under
  - Union
  - Intersection
  - Difference
  - Concatenation
  - Star Closure
  - Complementation
  - Reversal

  operations

Homomorphism

- Suppose Σ and Γ are alphabets, the function $h : \Sigma \rightarrow \Gamma^*$ is called a homomorphism
- It is a substitution in which a single symbol $a \in \Sigma$ is replaced by a string $x \in \Gamma^*$, that is. $h(a) = x$
- Extend to strings: $h(\omega) = h(a_1)\ldots, h(a_n)$ where $\omega \in \Sigma^*$ and $a_i \in \Sigma$
- Extend to languages $h(L) = \{h(\omega)|\omega \in L\}$
  - $h(L)$ is called the homomorphic image of L.

Homomorphism Example

- Let $\Sigma = \{a, b\}$ and $\Gamma = \{a, b, c\}$
  - $h(a) = ab$ and $h(b) = bbc$
  - $h(aba) = abbbcab$

Theorem

Let h be a homomorphism. If L is regular then $h(L)$ is also regular.

Proof

Obvious: Modify the DFA transitions

Decision Properties of Regular Languages

Theorem

Given a standard representation (DFA, NFA, RE) of any regular language L on Σ and any ω in Σ^*, there exists an algorithm to determine if ω is in L or not.

Proof.

Represent the language with a DFA and test if ω is accepted or not $\square$

Decision Properties of Regular Languages

Theorem

There exist algorithms for determining whether a regular language in standard representation is empty or not.

Proof.

Represent the language with a DFA. If there is a path from the start state to some final state, the language is not empty. □

Decision Properties of Regular Languages

Theorem

There exist algorithms for determining whether a regular language in standard representation is finite or infinite.

Proof.

Find all states that form a cycle. If any of these are on path from the start state to a final state, then the language is infinite. □

Proof.

If DFA with n states accepts some string of length between n and $2n - 1$ then it accepts an infinite set of strings.(needs Pumping Lemma) □

Decision Properties of Regular Languages

Theorem

Given standard representations of two regular languages L_1 and L_2, there exists an algorithm to determine if $L_1 = L_2$.

Proof.

Compute $L_3 = (L_1 - L_2) \cup (L_2 - L_1)$ which has to be regular. If $L_3 = \Phi$ then $L_1 = L_2$. □

More Decision Problems

- To decide if $L_1 \subseteq L_2$, check if $L_1 - L_2 = \Phi$
- To decide if $\epsilon \in L$, check if $q_0 \in F$
- To decide if L contains ω such that $\omega = \omega^R$
  - Let M be the DFA for L. Construct M^R.
  - Construct $M \cap M^R$ using the cross-product construction
  - Check if $L(M \cap M^R) \neq \Phi$.