Getting Formal

A finite automaton M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

Q is the finite set of states

Σ is the finite alphabet of symbols

$\delta: Q \times \Sigma \rightarrow Q$ is the state transition ***function***

q_0 is the start state

F is the set of accept states

A Convenient Definition

Suppose we are given a DFA $M = (Q, \Sigma, \delta, q_0, F)$

In state q, if the next input symbol is a, the next state is $\delta(q, a)$.

What is the final state when we start in q and process string w?

$$\Delta(q, w)$$

Define: $\Delta: Q \times \Sigma^* \rightarrow Q$ recursively as follows:

$$\Delta(q, \epsilon) = q$$

$$\Delta(q, ax) = \Delta(\delta(q, a), x), \qquad a \in \Sigma, x \in \Sigma^*$$

$$M \; accepts \; w \; iff \; \Delta(q_0, w) \in F$$

Terminology

The ***language*** of finite automaton M is the set of strings accepted by M.

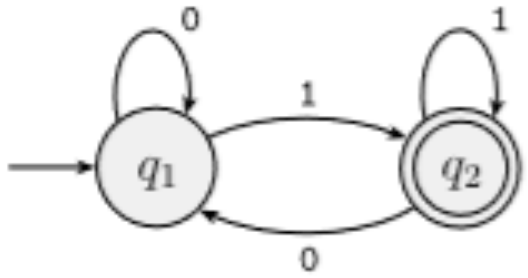
$$L(M) = \{w: M\ accepts\ w\}$$

We say that M ***recognizes*** A if $L(M) = A$.

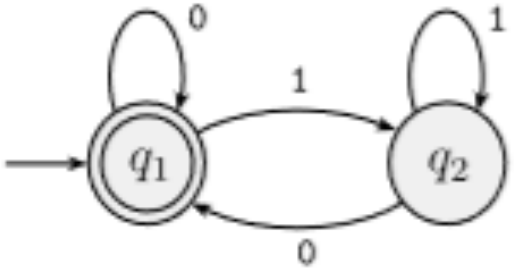
A language A is ***regular*** if it is recognized by a finite automaton.

Examples

What is the language of the following machines?



$L(M_1) = \{w: w \; ends \; in \; 1\}$



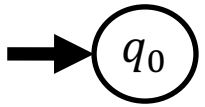
$L(M_2) = \{w: w \; ends \; in \; 0\}$?

But what about the empty string ϵ?

$L(M_2) = \{w: w = \epsilon \; or \; w \; ends \; in \; 0\}$

Designing Finite Automata

Problem: Design an FSA for the language $\{w: w\ ends\ in\ 000\}$

Begin with a start state: 

Next, pretend that you are the FSA!

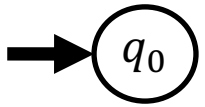
The input string is revealed one symbol at a time.
When you see a symbol you must decide the next state.
You must be in an accept state ***iff*** the entire input string has been processed.

Question: should the start state be an accepting state?

NO! Because then it will accept ϵ which definitely does ***not*** end in 000.

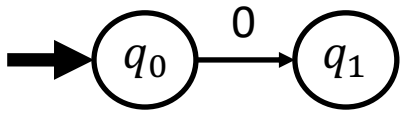
Designing Finite Automata

Problem: Design an FSA for the language $\{w: w\ ends\ in\ 000\}$



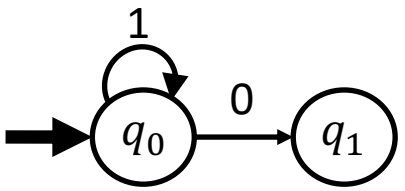
Question: Should I remain in the start state if I see a 0?

NO! Because if the input is 000 then I'm stuck in the rejecting start state



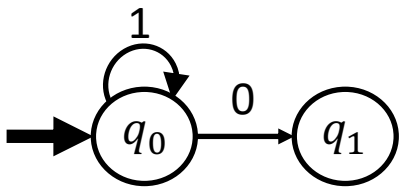
Question: What if the first input symbol was a 1?

Might as well stay put because 1 doesn't advance towards the goal of ending in 000.

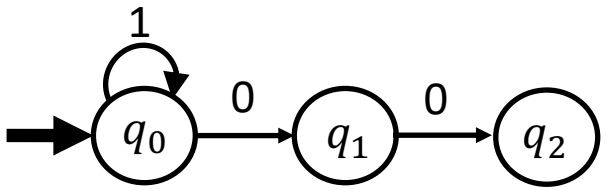


Designing Finite Automata

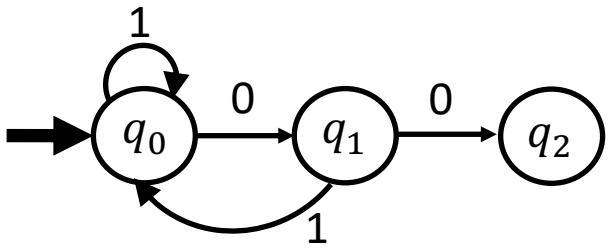
Problem: Design an FSA for the language $\{w: w\ ends\ in\ 000\}$



In q_1 if next symbol is 0:



In q_1 if next symbol is 1:

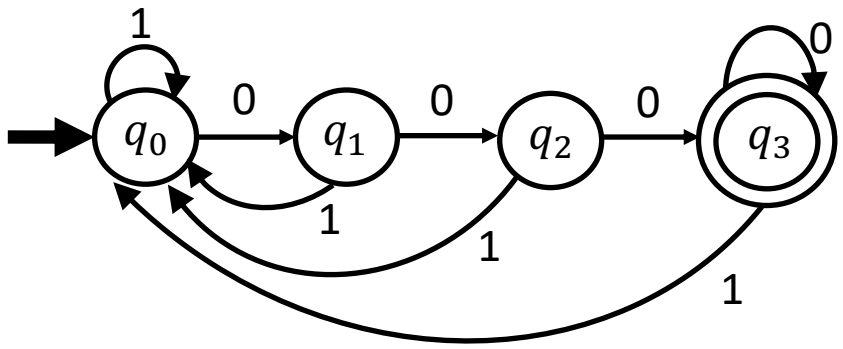


Designing Finite Automata

Problem: Design an FSA for the language $\{w: w\ ends\ in\ 000\}$

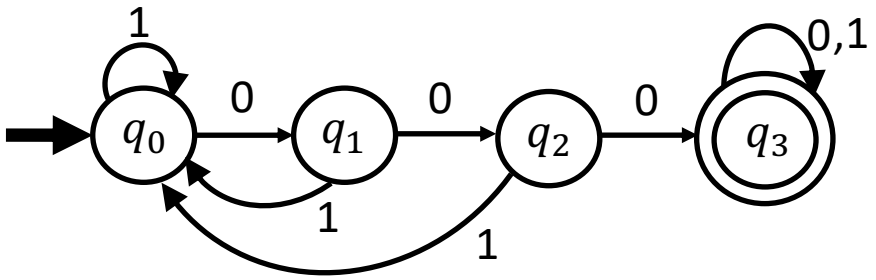
In q_2 if next symbol is 0: better accept in case this is the end of input

In q_2 if next symbol is 1:



Designing Finite Automata

Problem: Design an FSA for the language $\{w: w\ contains\ 000\}$



Searching for a pattern in a text string

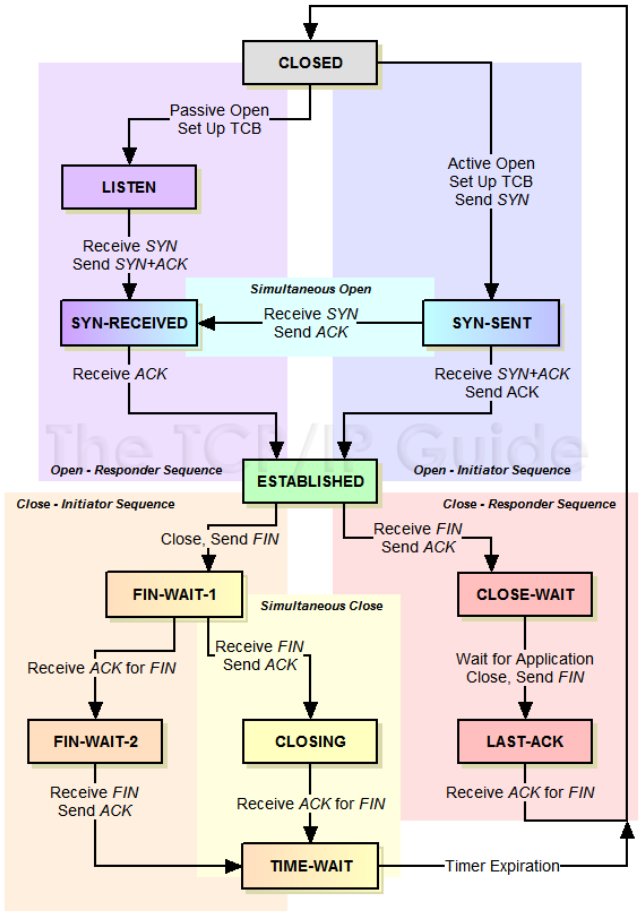
Find all occurrences of “Ali Bashir” or “Colin Creevey” in the text of Harry Potter and the Goblet of Fire.

Step 1: Design an FSA to recognize the language {Ali Bashir, Colin Creevey}

Step 2: Run the text of Harry Potter and the Goblet of Fire (the input string) through the FSA. Each time you enter an accept state, print the position in the text.

FSA theory is the foundation of pattern search and parsing.

More FSA Applications



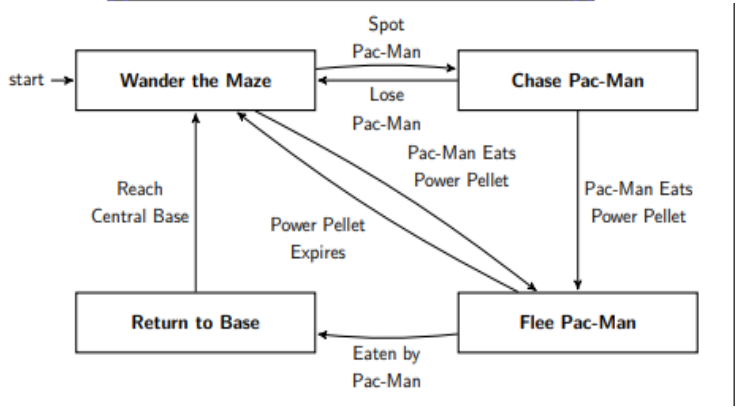
Protocol Specification and Analysis

Speech Recognition

Natural Language Processing

Automated Answering Services

Industrial Control Systems



Game Design

Regular Operations on Languages

Let A, B be two languages.

UNION $A \cup B = \{w: w \in A \; or \; w \in B\}$

CONCATENATION $A \circ B = \{xy: x \in A \; and \; y \in B\}$

STAR $A^* = \{x_1 x_2 \ldots x_k: \;\; k \geq 0 \; and \; \forall i \; x_i \in A\}$

concatenate any number of strings in A

An Example

$A = \{oreo, ginger\}$

$B = \{cookie, icecream\}$

UNION $A \cup B = \{oreo, ginger, cookie, icecream\}$

CONCATENATION

$A \circ B = \{oreocookie, oreoicecream, gingercookie, gingericecream\}$

STAR

$A^* = \{\epsilon, oreo, ginger, oreooreo, oreoginger, gingerginger, gingeroreo, oreooreooreo, \ldots\}$

Our First Theorem

Suppose that A, B are both regular languages.

What about $A \cup B$?

Theorem 1.25 The class of regular languages is closed under the union operation.

In other words, if A, B are both regular languages then $A \cup B$ is regular as well.

Why might this be useful?

In designing an DFA for language L it's sometimes easier to express $L = A \cup B$,

Next design separate DFAs for A and B, and

Finally combine the two DFAs to make a DFA for L.

Our First Theorem

Theorem 1.25 The class of regular languages is closed under the union operation.

Proof Idea

Let A, B each be a regular language

Since A is regular, let M_1 be a DFA that recognizes A.

Similarly, let M_2 be a DFA that recognizes B.

We'll construct DFA M for $A \cup B$ by using M_1, M_2.

Idea 1: Run the input string w through M_1 and then through M_2; accept w if either one accepts.

Problem: We only get to process each input symbol once – so, if M_1 rejects ….. 

Another Idea

Theorem 1.25 The class of regular languages is closed under the union operation.

Proof Idea

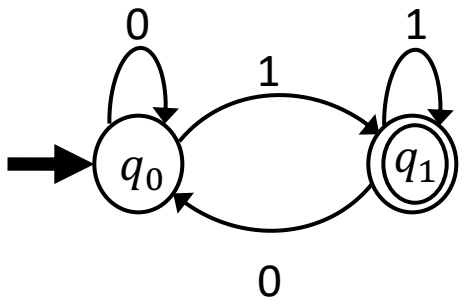
Idea 1: Run the input string w through M_1 and then through M_2; accept w if either one accepts.

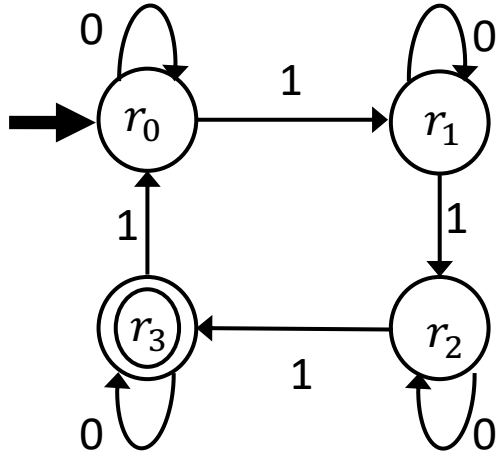
Problem: We only get to process each input symbol once – so, if M_1 rejects …..

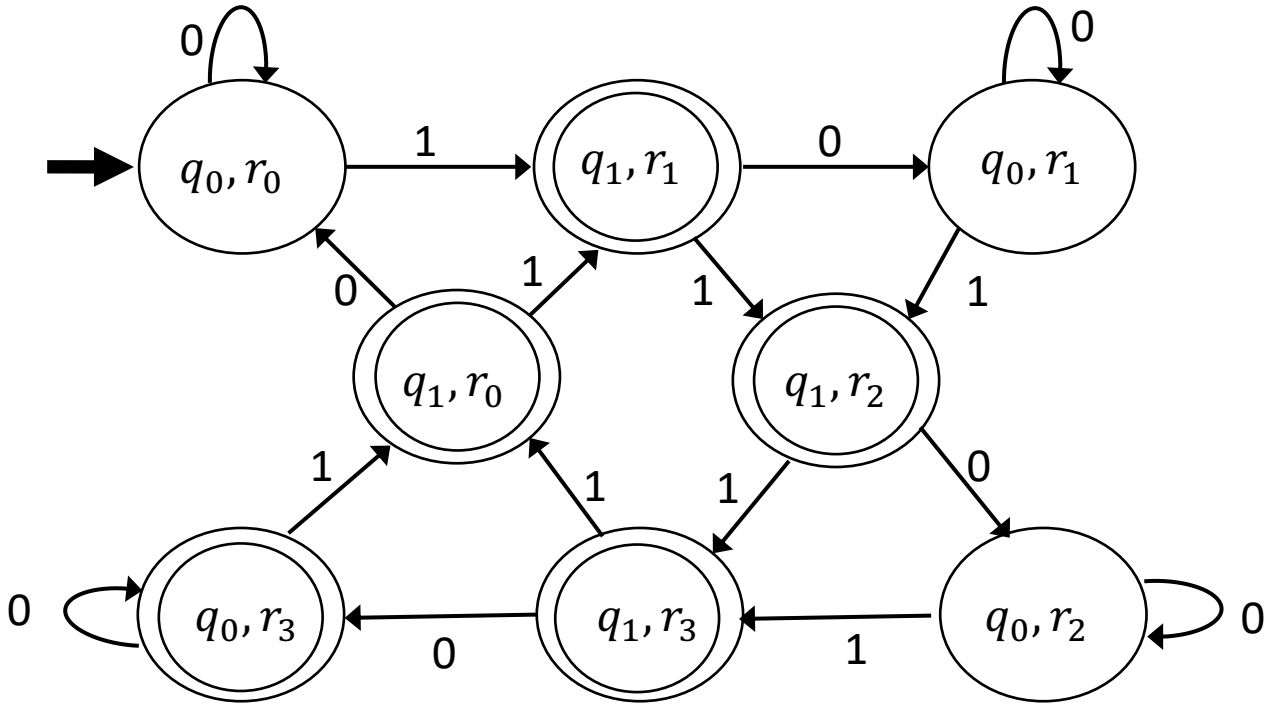
Idea 2: Run the input string simultaneously through M_1 and M_2; accept if either one accepts.

Question: But how???

How to simultaneously simulate two DFAs







Proof by Construction

Theorem 1.25 The class of regular languages is closed under the union operation.

Proof: Let A, B be regular languages recognized by DFAs M_1 and M_2.

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Define a new DFA $M = (Q, \Sigma, \delta, q_0, F)$:

For simplicity we'll assume that both machines have the same alphabet.

In homework you will modify the proof to remove this assumption.

$Q = Q_1 \times Q_2$ each state of M is a pair of states, one from M_1, and the other from M_2

$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$ a transition in M tracks transitions in both M_1, M_2!!!

$q_0 = (q_1, q_2)$ start M in the start states of M_1, M_2

$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ accept if one of the machines ends in an accept state!

A formal proof of correctness proceeds by induction on the length of the input string.

Closure under Concatenation

Theorem 1.26 The class of regular languages is closed under the concatenation operation.

Let A, B be regular languages recognized by DFAs M_1 and M_2.

How do we construct M to recognize $A \circ B$?

Input string w should be recognized *iff* $w = xy$ where $x \in A \; and \; y \in B$.

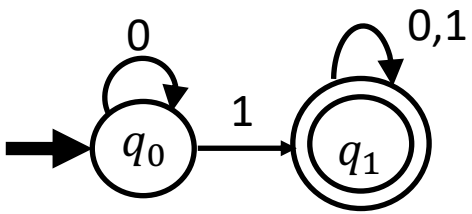
Idea 1. Start processing w through M_1

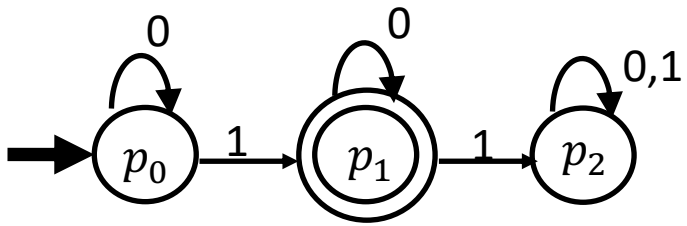
When x has been processed, and if M_1 is in an accept state,

proceed to the start state of M_2 to process y.

An Attempt

$A = \{w: w\ contains\ one\ or\ more\ 1's\}$ $B = \{w: w\ contains\ one\ 1\}$





$A \circ B = \{xy: x\ contains\ one\ or\ more\ 1's\ and\ y\ contains\ one\ 1\}$

When do we transit from $q_1\ to\ p_0$?

How do know that there is exactly one 1 in the rest of the (unseen) input?

How does one even transit from $q_1\ to\ p_0$?

All transitions from each state are already accounted for!

An Impossible Challenge?

We can use M_1, M_2 to recognize x and y

But we don't know what x is – so when do we switch over to M_2?

And what if we switch over to M_2 at the wrong time?

And what about the fact that M_1, M_2 cannot accommodate new transitions?

What do older siblings do when they're losing a game to their younger sibling?

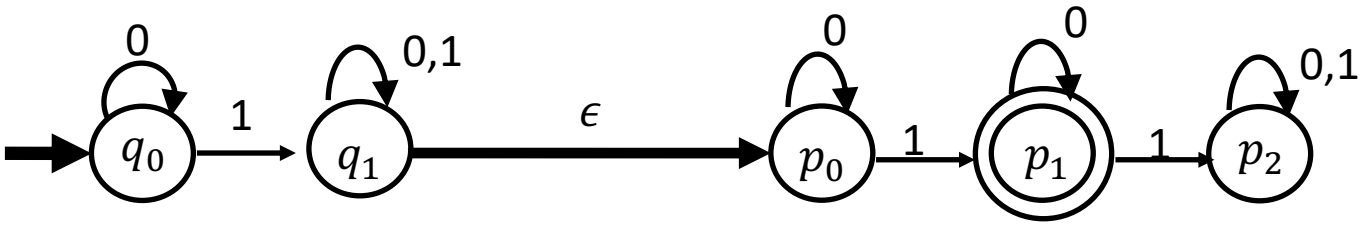
Change the rules of the game!

How to Cheat and Win!

Insert a transition labeled ϵ from the $M_1's$ accept state to $M_2's$ start state.

The machine can magically guess when to take this transition (free move).

How does this solve the problem?



Magically guess when one 1 remains in the unseen input: now take the ϵ transition!

Starting from q_1 the next state on reading a 0 is either $q_1 or\ p_0$!