Ex. Show $L = \{0^n 1^n | n \geq 0\}$ is *not* regular.

Suppose it is. By PL $\exists p$. Consider $s = 0^p 1^p = xyz$. Since $|xy| \leq p$, $y \neq \varepsilon$, $y = 0^j$, $j > 0$. And $xy^2z = 0^{p+j}1^p \in L$, which is a contradiction.

Ex. Show $L = \{1^p |\ p \text{ is prime }\}$ is not regular.

Suppose it is. By PL $\exists n$. Consider some prime $p \geq n + 2$.

Let $1^p = xyz$, $|y| = m > 0$. So $|xz| = p - m$.

Consider $xy^{(p-m)}z$ which must be in L.

But
$|xy^{(p-m)}z| = |xz| + |y|(p-m) = (p-m) + m(p-m) = (p-m)(1+m)$

Now $1 + m > 1$ since $y \neq \varepsilon$, and $p - m > 1$ since $p > n + 2$ and $m = |y| \leq |xy| \leq n$. So the length of $xy^{(p-m)}z$ is not prime, and hence it cannot be in L — contradiction.

Closure Properties of Regular Languages

Union: If L, M are regular, so is $L \cup M$.

Proof: $L = L(R)$ and $M = L(S)$, so $L \cup M = L(R + S)$.

Complementation: If L is regular, so is $L^c = \Sigma^* - L$.

Proof: $L = L(A)$, so $L^c = L(A')$, where A' is the DFA obtained from A as follows: $F_{A'} = Q - F_A$.

Intersection: If L, M are regular, so is $L \cap M$.

Proof: $L \cap M = \overline{\overline{L} \cup \overline{M}}$.

Reversal: If L is regular, so is $L^R = \{w^R | w \in L\}$, where $(w_1 w_2 \dots w_n)^R = w_n w_{n-1} \dots w_1$.

Proof: Given a reg exp E, define E^R by structural induction. The only trick is that $(E_1 E_2)^R = E_2^R E_1^R$.

Homomorphism: $h : \Sigma^* \longrightarrow \Sigma^*$, where
$h(w) = h(w_1 w_2 \ldots w_n) = h(w_1)h(w_2)\ldots h(w_n)$.

Ex. $h(0) = ab, h(1) = \varepsilon$, then $h(0011) = abab$.

$h(L) = \{h(w)|w \in L\}$

If L is regular, then so is $h(L)$.

Proof: Given a reg exp E, define $h(E)$.

Inverse Homomorphism: $h^{-1}(L) = \{w|h(w) \in L\}$.

Proof: Let A be the DFA for L; construct a DFA for $h^{-1}(L)$ as follows: $\delta(q, a) = \hat{\delta}_A(q, h(a))$.

Complexity of converting among representations

ε-NFA $\longrightarrow$ DFA is $O(n^3 2^n)$
$O(n^3)$ for computing the ε closures of all states – Warshall's algorithm, and 2^n states

DFA $\longrightarrow$ NFA is $O(n)$

DFA $\longrightarrow$ Reg Exp is $O(n^3 4^n)$
There are n^3 expressions $R_{ij}^{(k)}$, and at each stage the size quadruples (as we need four stage $(k-1)$ expressions to build one for stage k)

Reg Exp $\longrightarrow$ ε-NFA is $O(n)$
The trick here is to use an efficient parsing method for the reg exp; $O(n)$ methods exist

Decision Properties

1. Is a language empty?

   Automaton representation: Compute the set of reachable states from q_0. If at least one accepting state is reachable, then it is not empty.

   What about reg exp representation?

2. Is a string in a language?

   Translate any representation to a DFA, and run the string on the DFA.

3. Are two languages actually the same language?

   Equivalence and minimization of Automata.

Equivalence and Minimization of Automata

Take a DFA, and find an *equivalent* one with a *minimal* number of states.

Two states are equivalent iff for all strings w,

$$\hat{\delta}(p, w) \text{ is accepting} \iff \hat{\delta}(q, w) \text{ is accepting}$$

If two states are not equivalent, they are *distinguishable*.

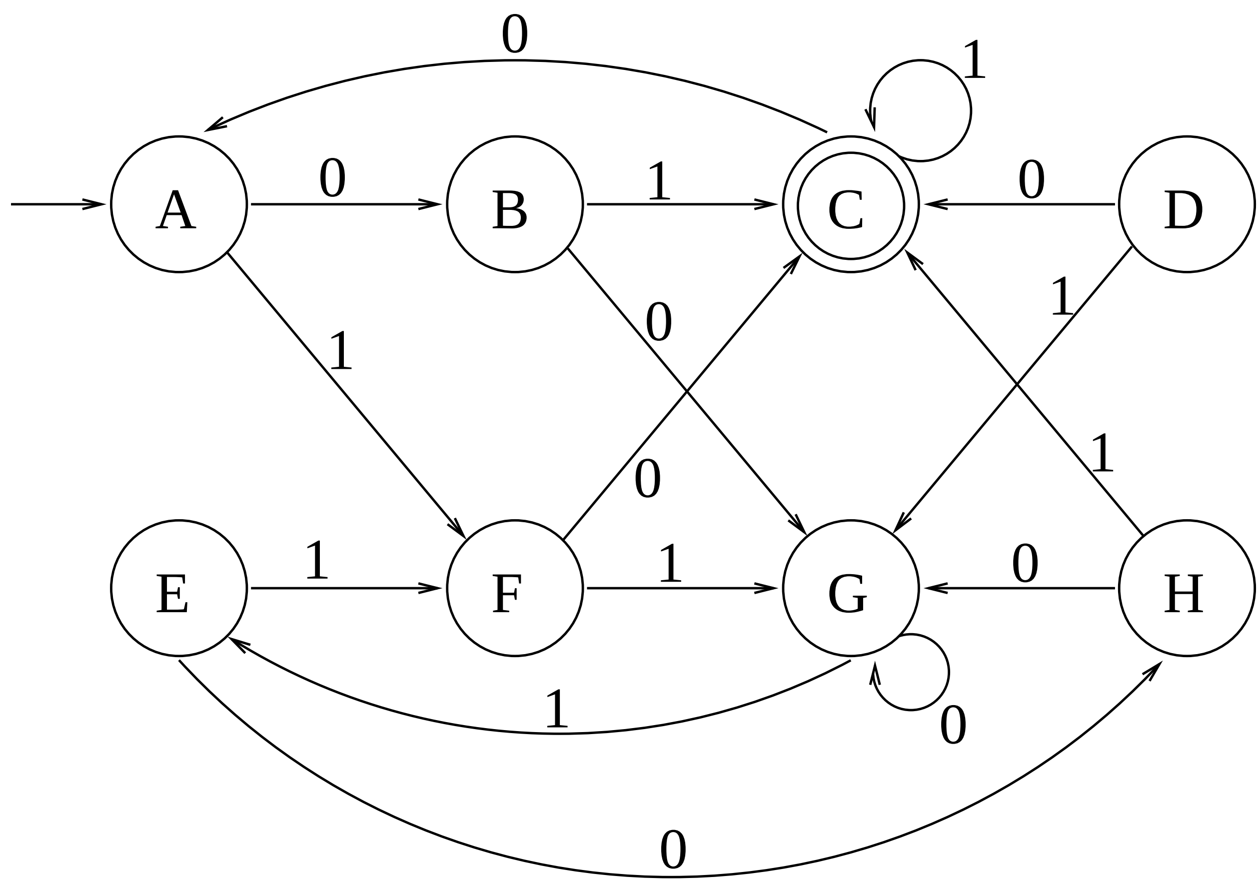
Find pairs of distinguishable states: Basis Case: if p is accepting and q is not, then $\{p, q\}$ is a pair of distinguishable states.

Induction Step: if $r = \delta(p, a)$ and $s = \delta(q, a)$, where $a \in \Sigma$ and $\{r, s\}$ are distinguishable, then $\{p, q\}$ are distinguishable.

Table Filling Algorithm

A recursive algorithm for finding distinguishable pairs of states.

Ex.



Solution:

	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Distinguishable states are marked by "x"; the table is only filled below the diagonal (above is symmetric).

Theorem: If two states are not distinguished by the algorithm, then the two states are equivalent.

Proof: Use the Least Number Principle (LPN): any set of natural numbers has a least element.

Let $\{p,q\}$ be a distinguishable pair, for which the algorithm left the corresponding square empty, and furthermore, of all such "bad" pairs $\{p,q\}$ has a shortest distinguishing string w.

Let $w = a_1a_2 \ldots a_n$, $\hat{\delta}(p,w)$ is accepting & $\hat{\delta}(q,w)$ isn't.

$w \neq \varepsilon$, as then p,q would be found out in the Basis Case of the algorithm.

Let $r = \delta(p,a_1)$ and $s = \delta(q,a_1)$. Then, $\{r,s\}$ are distinguished by $w' = a_2a_3 \ldots a_n$, and since $|w'| < |w|$, they were found out by the algorithm.

But then $\{p,q\}$ would have been found in the next stage.

Equivalence of DFAs

Suppose D_1, D_2 are two DFAs. To see if they are equivalent, i.e., $L(D_1) = L(D_2)$, run the table-filling algorithm on their "union", and check if $q_0^{D_1}$ and $q_0^{D_2}$ are equivalent.

Complexity of the Table Filling Algorithm: there are $n(n-1)/2$ pairs of states. In one round we check all the pairs of states to check if their successor pairs have been found distinguishable; so a round takes $O(n^2)$ many steps. If in a round no "x" is added, the procedure ends, so there can be no more than $O(n^2)$ rounds, so the total running time is $O(n^4)$.