Predicate:
a mathematical predicate can be True or False.
predicates are functions with boolean return values:
P, Q(n), R(x,y,z)

Logical Operators:
AND: $P \land Q$ || OR: $P \lor Q$ || NOT: $\neg P$
Implication: $P \implies Q = \neg P \lor Q$

Distributive Rule
$P \land (Q \lor R) = (P \land Q) \lor (P \land R)$
$P \lor (Q \land R) = (P \lor Q) \land (P \lor R)$

De Morgans Law
$\neg(P \land Q) = \neg P \lor \neg Q$
$\neg(P \lor Q) = \neg Q \implies \neg P$
$P \implies Q = \neg Q \implies \neg P$

Implication
$P \implies Q = True \;\&\; \neg P \implies Q = True$
$P \implies \neg Q = False$
$\neg P \implies \neg Q = True$

Quantors
OR: $\bigvee_{k=0}^{n} P_k$
P true for any $k \in 0..n$

AND: $\bigwedge_{k=0}^{n} P_k$
P true for all $k \in n$

All: $\forall k \in 0..n = P_k$
for all k P = True

Exists: $\exists k \in 0..n = P_k$
a k exists where P = True

Normalforms
disjunctive
$(x1 \land x2) \lor (\overline{x1} \land x2) \lor (x1 \land \overline{x2})$

conjunctive
$(x1 \lor x2) \land (\overline{x1} \lor x2) \land (x1 \lor \overline{x2})$

These are useful for true and false tables
This one would result to true if x1 or x2 is true.

Quantities:
$\emptyset = \{\}$ $\quad [n] = \{0..n\}$ $\quad$ {a .. z}

Union:
$A \cup B = \{x | x \in A \lor x \in B\}$

Intersection:
$A \cap B = \{x | x \in A \land x \in B\}$

Complement:
$\overline{A} = \{x | x \notin A\}$

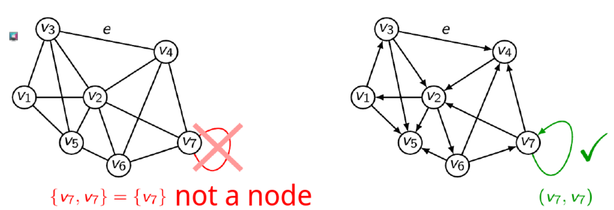
Difference:
$A \setminus B = \{x \in A | x \notin B\}$

Pairs $A \times B = \{(a, b) | a \in A \land b \in B\}$

n-Tuples $\bigtimes_{k=0}^{n} A_i = \{(a_o, a_i, .., a_n) | a_i \in A_i\}$

Undirected Graph
doesn't have directions, and therefore can't have edges to itself

Directed Graph
This does have directions, therefore an edge to itself is valid!



Vertices V = $\{v_1, v_2, ..., v_n\}$ Edge e = $\{v_3, v_4\}$
EdgeCount E = $\{e | e Edge\}$

Proofs:
constructive Proof (proof by reforming)
Consider $ax^2 + bx + c = 0$, we can proof this to have 2 solutions by reforming.

$$ax^2 + bx + c = 0$$
$$x^2 + 2\frac{b}{2a}x + \frac{b^2}{4a^2} - \frac{b^2}{4a^2} + \frac{c}{a} = 0$$
$$x^2 + 2\frac{b}{2a}x + \frac{b^2}{4a^2} = \frac{b^2 - 4ac}{4a^2}$$
$$\left(x + \frac{b}{2a}\right)^2 = \frac{b^2 - 4ac}{4a^2}$$
$$x + \frac{b}{2a} = -\frac{b}{2a} \pm \sqrt{\frac{b^2 - 4ac}{4a^2}}$$

If $b^2 - 4ac > 0$ then we have 2 solutions!

Proof by contradiction
Take -2, it isn't a natural number. We can prove this by claiming the opposite.
If -2 is a natural number, then it has all the attributes of a natural number.
For example, it should be possible to take the square root of -2.
$\sqrt{-2} = NaN$
As you can see -2 does not have this attribute and is therefore
NOT a natural number!

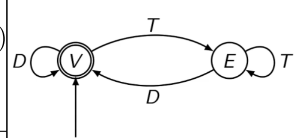
Proof by Induction
This is particularly useful if you want to check an attribute
for a range of numbers such as n or n+1

Base claim:
$$P(n) = \sum_{k=1}^{n} = \frac{n(n+1)}{2}$$
Anker: check for n=1
$$P(1) = \frac{1(1+1)}{2} = 1$$

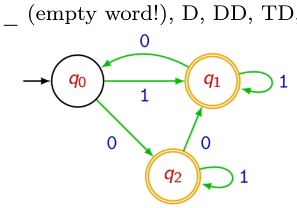
Hypothesis: it also works for n+1
$$P(n+1): \sum_{k=1}^{n+1} k = \left(\sum_{k=1}^{n} k\right) + n + 1 = \frac{(n+1)(n+2)}{2} = \frac{(n+1)(n+1+1)}{2} \checkmark$$

Alphabet and Word
Σ = Alphabet: Nonempty Quantity of characters
$\Sigma^n = \Sigma \times ... \times \Sigma$ = String
$w \in \Sigma^n$ An element in that string is a Word with length n.
$\varepsilon \in \Sigma^0$ The empty word, don't forget the empty word!
Quantity of all words:
$\Sigma^* = \{\varepsilon\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup ... = \bigcup_{k=0}^{\infty} \Sigma^k$

Language $L \subset \Sigma^*$ = Language
$L = \emptyset \subset \Sigma^*$ = Empty Language
$L = \Sigma^* \to \Sigma = \{0, 1\}$ all binary strings.
A language is regular if a DFA can be formed out of it.

Deterministic Finite Automaton (DFA/DEA)


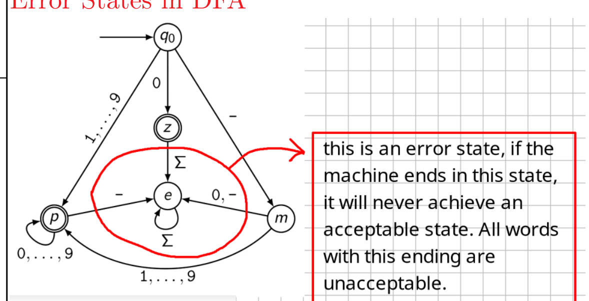
A very simple machine that accepts a variety of inputs.
Only requirement is that a D follows after T.
This means all the following inputs are valid:
_ (empty word!), D, DD, TD, TTTTTTTD, DDDDDTD, DDDDD, ...

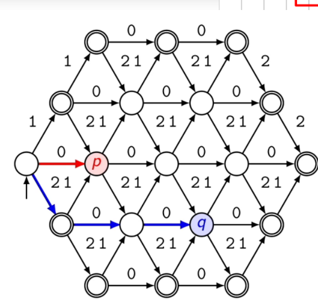


Machine A: $\{Q, \Sigma, \delta, q_0, F\}$
State = Q -> $\{q_1, q_2, ..., q_n\}$
Alphabet = Σ
Transitioning-Function = $\delta : Q \times \Sigma \to Q$
Starting State = $q_0 = L(\varepsilon) = L$
Acceptable Endstates = $F \subset Q$

		0	1	
all states	Q			Σ possible inputs, here 0 or 1
	q_0	q_2	q_1	
all acceptable states. F	q_1 /F	q_0	q_1	δ current position. -> change to q1
	q_2 /F	q_1	q_2	

Language of DFA A:
$L(A) = \{w \in \Sigma^* | A \text{ accepts } w\} = \{w \in \Sigma^* | \delta(q_0, w) \in F\}$
The language of a DFA is simply all accepted words!

Error States in DFA




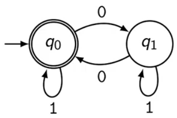
from q, many paths lead to F
This means 11, or 0 would be the "same"
$L(q) = \{0, 10, 11, 12, ...\}$
The same would obviously apply to P

Myhill-Nerode

Adding a word to a word, to make it compatible with a language

$L(w) = \{w' | ww' \in L\}$ including: $L(\varepsilon)$!

w	$L(w)$	Q
ε	$L(\varepsilon) = L$	q_0
0	$L(0) = \{w \in \Sigma^* \mid \|w\|_0 \text{ uneven } \}$	q_1
1	$L(1) = \{w \in \Sigma^* \mid \|w\|_0 \text{ even } \} = L$	q_0
$\vdots$	$\vdots$	$\vdots$



even and uneven amounts of 0s

mod 2 zero's, 1's don't matter

Detecting Nonregular Languages with Myhill

The examples before always had a specific amount of words/characters
that one had to add, in order to accept the word.
However, there are languages that would need infinite states
in order to find the entire language of a DFA
A good example for this is the language 1^n0^n

w	$L(w)$	Q
ε	$\{0^n1^n \mid n \geq 0\}$	q_0
0	$\{0^n1^{n+1} \mid n \geq 0\}$	q_1
00	$\{0^n1^{n+2} \mid n \geq 0\}$	q_2
000	$\{0^n1^{n+3} \mid n \geq 0\}$	q_3
0^k	$\{0^n1^{n+k} \mid n \geq 0\}$	q_k
	$\vdots$	
01	$\{\varepsilon\}$	
001	$\{1\}$	
0001	$\{11\}$	
	$\vdots$	
1	$\emptyset$	e
10	$\emptyset$	e
	$\vdots$	

for every 0 that we add, we need a 1
this means that for n+k 0's we need k 1's
as $\lim_{k\to\infty}$ we need ∞ states!
not possible with a Deterministic automaton!

also note: we have clear error states
anything starting with 1 is an error.