Professor
Wayne Hayes

ICS 6B

Jan 12 – March 14, 2017
Momoko's note

Boolean Algebra & Logic

Jan 12, 2017 (TH)

Def **Proposition** ··· direct statement of fact (can be either true/false, but NOT both)
ex. "Tronto is the capital of Canada." → False

Def **$\neg p$** = "NOT p" = negation of p

TRUTH TABLE

p	$\neg p$
T	F
F	T

Def **$p \wedge q$** "and" "but" conjunction of p and q
↳ in English sentence, "but" is often used to show more than 1 event that occur simultaneously

Def **$p \vee q$** = "OR" disjunction

Def **$p \oplus q$** = "exclusive or" "XOR" (precisely one)

p	q	$p \wedge q$	$p \vee q$	$p \oplus q$
T	T	T	T	F
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F

$(\neg p \to q) \equiv p \vee q$

Def **$p \to q$** = "if p, then", "p implies q", "p only if q", "p is sufficient for q"
conditional statement aka logical implication

* this is opposite to causal implication
in logical implication, we find / see the result
and determine the fact (result → fact)
which is NOT fact → causing something
since q must be 100% (happens) unlike real world

p	q	$p \to q$	$\neg p$	$\neg p \to q$
T	T	T	F	T
T	F	F	F	T
F	T	T	T	T
F	F	T	T	F

if x doesn't exist,
the condition doesn't matter

Def **converse** of $p \to q$ is $q \to p$

Def **contrapositive** of $p \to q$ is $\neg q \to \neg p$, which is equivalent to the original statement

Def **inverse** of $p \to q$ is $\neg p \to \neg q$

Def **$p \leftrightarrow q$** = "if and only if" = biconditional statement of p and q
$\equiv (p \to q) \wedge (q \to p)$ aka "p is equivalent to q"

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Precedence of Logical Operators p.11

$\neg, \wedge, \vee, \to, \leftrightarrow$

ex. $\neg p \to q \to r$
① ② ③

* Logic and Bit Operations ··· check p.11

ex. Knight & Knave

a dishonest JAck in cards

knights always tell truth

knaves always tell a lie

A says "B is a knight".

B says "We are opposite type".

proposition … p = "A is a knight"

q = "B is a knight"

where to start

If p, then A says truth ("B is a knight"), so q is true

If q, then B says truth ("We are opposite types"), so p is false.

$(p \to q) \land (q \to \neg p)$

$p_{\neg p}$	q	$p \to q$	$q \to \neg p$	$(p \to q) \land (q \to \neg p)$
T_F	T	T	F	F
T_F	F	F	T	F
F_T	T	T	T	T
F_T	F	T	T	T

NOT TRUE

(premise : if p)

If $\neg p$, then A says a lie ("B is a knight"), so $\neg q$.

If $\neg q$, then B says a lie ("We are opposite type"), so $\neg p$.

$(\neg p \to \neg q) \land (\neg q \to \neg p)$

$p_{\neg p}$	$q_{\neg q}$	$\neg p \to \neg q$	$\neg q \to \neg p$	$(\neg p \to \neg q) \land (\neg q \to \neg p)$
T_F	T_F	F	T	F
T_F	F_T	T	F	F
F_T	T_F	F	T	F
F_T	F_T	T	T	T

← premise $(\neg p)$ is true

∴ A is a knave

and B is also a knave.

Jan 17, 2017

Def Bit = "Binary Digit" = T(1) / F(0)

	0	1	1	0	1	0	1
	1	1	0	1	0	0	1
AND	0	1	0	0	0	0	1
OR	1	1	1	1	1	0	1
XOR	1	0	1	1	1	0	0

check p.11

Def tautology is always true ex. $p \vee p$

p25

Def contradiction is always false ex. $p \wedge \neg p$

Def contengency can be true or false (neither tautology nor contradiction)

Def $p \equiv q$ = "triple equal" identically = p and q are logically equivalent if $p \leftrightarrow q$ is a tautology

p.27

p.28

TABLE 6 Logical Equivalences.

Equivalence	*Name*
$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity laws
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$	Negation laws

TABLE 7 Logical Equivalences Involving Conditional Statements.

$$p \to q \equiv \neg p \vee q$$
$$p \to q \equiv \neg q \to \neg p$$
$$p \vee q \equiv \neg p \to q$$
$$p \wedge q \equiv \neg(p \to \neg q)$$
$$\neg(p \to q) \equiv p \wedge \neg q$$
$$(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$$
$$(p \to r) \wedge (q \to r) \equiv (p \vee q) \to r$$
$$(p \to q) \vee (p \to r) \equiv p \to (q \vee r)$$
$$(p \to r) \vee (q \to r) \equiv (p \wedge q) \to r$$

TABLE 8 Logical Equivalences Involving Biconditional Statements.

$$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$$
$$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$$
$$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$$
$$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$$

Predicates & Quantifiers

p37

* Propositional logic deals with small fixed sets of objects

* We want to talk about sets of objects "for all" and "there exists" ← quantifiers

* We want variables in our expressions. ← predicates

e.x. predicate $P(x) = $ "$x>3$"

$P(4) = T$ (since $x=4>3$)

$P(2) = F$ (since $x=2 \not> 3$)

e.x. $Q(x,y) = $ "$x=y+3$"

$Q(6,3) = T$ (since $6=3+3$)

$Q(6,2) = F$ (since $6 \neq 2+3$)

e.x. consider the swap operation

Swap(x,y) : Pre $x=a \wedge y=b$

Post $x=b \wedge y=a$

where a & b are constant

```
sample code
swap(x,y)
  temp = x
  x = y
  y = x
end swap
```

predicate

Def $\exists x \in S \; Q(x)$ = "There exists an x in the set S such that Q(x)"

↳ existential quantifier = "there exists"

Def $\forall x \in S \; Q(x)$ = "For every x in the set S, Q(x)"

↳ universal quantifier = "for every x" "for all x"

* De Morgan's Laws for Quantifiers

p47

$\neg\exists x \, P(x) \equiv \forall x \neg P(x)$ $\quad$ $\neg\forall x \, P(x) \equiv \exists x \neg P(x)$

* Combining Quantifiers

p.60

… when mixing, order is important

e.g. $\forall x \exists y \, P(x,y) \not\equiv \exists y \forall x \, P(x,y)$

such like $P(x,y) = $ "$x+y=0$" …

$\forall x \exists y \, P(x,y)$ = "For every real number x, there is a real number y s.t. P(x,y) ← True

$\exists y \forall x \, P(x,y)$ = "There is a real number y s.t. for every real number x, P(x,y) ← False

such like $P(x,y) = $ "$x+y=z$"

$\forall x \forall y \exists z \, P(x,y,z)$ … True

$\exists z \forall x \forall y \, P(x,y,z)$ … False (There is no magic number z whose value is the sum of any x and any y)

but if $P(x,y) = $ "$x+y=y+x$", it work ← sometimes it works, depends on P(x,y)

← go through all possibility in your head and see if it's true or not

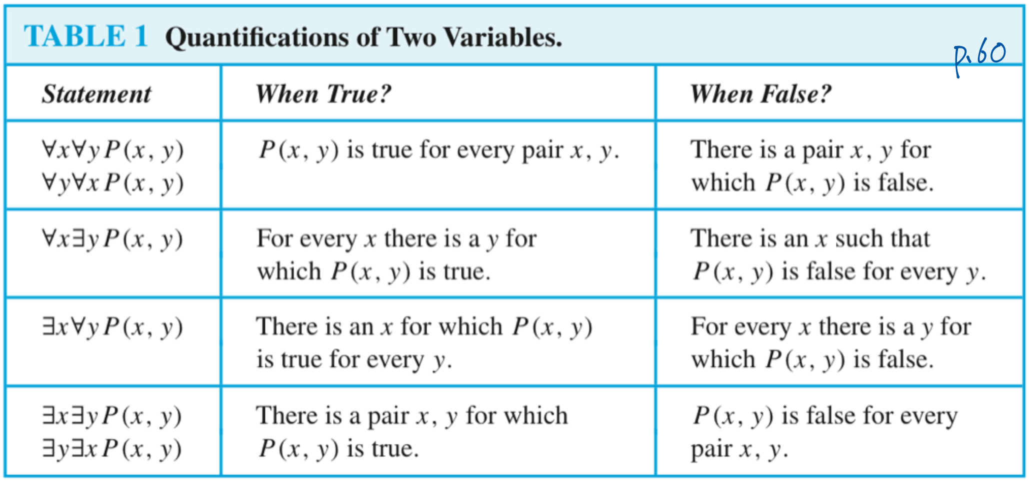


TABLE 1 Quantifications of Two Variables.

p.60

Statement	*When True?*	*When False?*
$\forall x \forall y P(x,y)$ $\forall y \forall x P(x,y)$	$P(x,y)$ is true for every pair x, y.	There is a pair x, y for which $P(x,y)$ is false.
$\forall x \exists y P(x,y)$	For every x there is a y for which $P(x,y)$ is true.	There is an x such that $P(x,y)$ is false for every y.
$\exists x \forall y P(x,y)$	There is an x for which $P(x,y)$ is true for every y.	For every x there is a y for which $P(x,y)$ is false.
$\exists x \exists y P(x,y)$ $\exists y \exists x P(x,y)$	There is a pair x, y for which $P(x,y)$ is true.	$P(x,y)$ is false for every pair x, y.

*Practice: Translate from English to Logic & from Logic to English

- "the sum of two positive integers is positive"

  $= \forall x \forall y \in \mathbb{Z} \quad x>0 \wedge y>0 \Rightarrow x+y=0$ ← your $P(x,y)$ ←Note: there are many ways to say

- $C(x)$ = "x owns a computer"

  $F(x,y)$ = "x and y are friends"

  $\forall x\, (C(x) \vee \exists y\, (C(y) \wedge F(x,y)))$

  ∴ everyone has a computer or a friend who has a computer

- $\exists x \forall y \forall z\, ((F(x,y) \wedge F(x,z) \wedge (y \neq z)) \rightarrow \neg F(y,z))$

  Step 1 examine $(F(x,y) \wedge F(x,z) \wedge (y \neq z)) \rightarrow \neg F(y,z)$

  ∴ if student x and y are friends, x and z are friends, and if y and z are not the same student,
  then y and z are not friends

  Step 2 original statement ∴ there is a student x s.t. for all students y and all students z other then y
  if x and y are friends and if x and z are friends, then y and z are not friends

  Step 3 generalize the expression ∴ "there is a student none of whose friends are also friends with each other"

- "There is a woman who has taken a flight on every airline in the world."

  Step 1 change the statement into more "logical way"

  ∴ "There is a woman on the Earth s.t. for every airline on the Earth
  and there is a flight of that airline that the woman has taken"

  Step 2 create the proposition

  $T(w,f)$ = "w has taken flight f"

  $S(f,a)$ = "f is a scheduled flight (route) on airline a"

  Step 3 $\exists w \forall a \exists f\, (S(f,a) \wedge T(w,f))$

Rules of Inference ↙ a conclusion reached on the basis of evidence and reasoning

Jan 19, 2017

Def Argument is a sequence of statements that end with a conclusion

↙ it doesn't need to be true in general

Def Argument is valid if its conclusion (or final statement) follow from the truth of the preceding statements (premises) of the argument.

Def Fallacy is an invalid argument where tautology is surreptitiously replaced by contingency as if the contingency were always true.

p.72

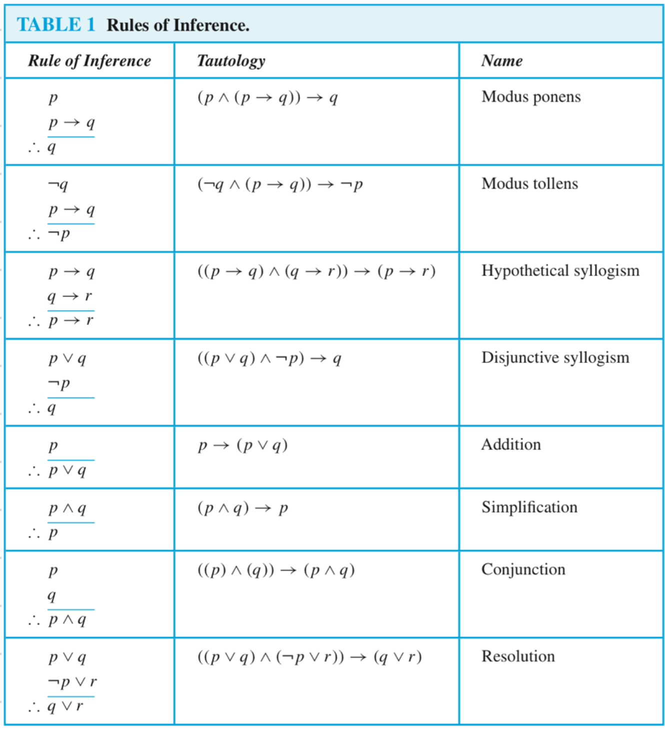


TABLE 1 Rules of Inference.

Rule of Inference	*Tautology*	*Name*
p $p \to q$ $\therefore q$	$(p \land (p \to q)) \to q$	Modus ponens
$\neg q$ $p \to q$ $\therefore \neg p$	$(\neg q \land (p \to q)) \to \neg p$	Modus tollens
$p \to q$ $q \to r$ $\therefore p \to r$	$((p \to q) \land (q \to r)) \to (p \to r)$	Hypothetical syllogism
$p \lor q$ $\neg p$ $\therefore q$	$((p \lor q) \land \neg p) \to q$	Disjunctive syllogism
p $\therefore p \lor q$	$p \to (p \lor q)$	Addition
$p \land q$ $\therefore p$	$(p \land q) \to p$	Simplification
p q $\therefore p \land q$	$((p) \land (q)) \to (p \land q)$	Conjunction
$p \lor q$ $\neg p \lor r$ $\therefore q \lor r$	$((p \lor q) \land (\neg p \lor r)) \to (q \lor r)$	Resolution

premises conclusion

e.g. $\underbrace{[(p\to q)\land q]}_{\text{statement ①}} \to \underbrace{p}_{②}$

This is a contengency, not a tautology

"fallacy of affirming the conclusion"

(you cannot conclude it)

if p="it's raining"
q="there is a cloud"
rain implies the existence of clouds, but the entire statement is a contengency

e.g. $[(p\to q)\land \neg p]\to \neg q$ p.75

"fallacy of denying the hypothesis"

*Inference in Quantified Statements

- universal instantiation

  $(\forall x\in S\ P(x))\to P(c)$ for any individual $c\in S$

  e.g. "All humans are mortal" ← where M(x)="x is mortal"

  "Socrates is a human" H={all humans}

  ⇒ "Socrates is mortal" $(\forall x\in H\ M(x)\ \land\ s\in H)\to M(s)$

- extential instantiation

  $\exists x\in S\ P(x)$, assume c is one such element s.t. P(c)

  we don't necessarily know the value but we know it exists, so we name it c and continue our argument

- existentional generalization

  conclude $\exists x P(x)$ when there is $c\in S$ s.t. P(c) is true

- universal generalization

  if P(c) is true for all (arbitrary element c), $\forall x P(x)$ is true

p.76

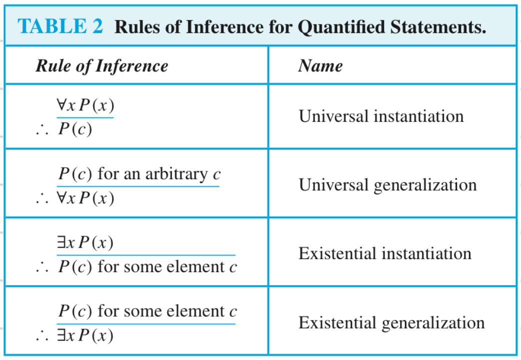


TABLE 2 Rules of Inference for Quantified Statements.

Rule of Inference	*Name*
$\forall x P(x)$ $\therefore P(c)$	Universal instantiation
$P(c)$ for an arbitrary c $\therefore \forall x P(x)$	Universal generalization
$\exists x P(x)$ $\therefore P(c)$ for some element c	Existential instantiation
$P(c)$ for some element c $\therefore \exists x P(x)$	Existential generalization

*Combining Rules of Inference for Propositions & Quantified Statement p.77

universal modus ponents

$\forall x\in S\ (P(x)\to Q(x))$

$a\in S$

$\underline{P(a)}$

$Q(a)$

Introduction to Proofs p.80

there is a difference between formal & informal proof
↳ like human convesation

Def Theorem is a statement that can be shown to be true (facts/results)
↳ a formed statement that has been proved correct

* Less important theorems sometimes are called propositions

Def We demonstrate that a theorem is true with a proof (a valid argument)

Def Axioms (postulates) are statements we assume to be true
↑ is it a principle?

Def A lemma is a "small theorem" which is often used to help prove a bigger theorem

Def A corollary is an immidiate (obvious) consequence of a just proved Thm

Def A conjecture is a statement believed to be true but not yet proved

Def Direct Proof uses sequence implications with axioms and previously proven statements
pure deduction

∴ mainly directly $p \to \cdots \to q$

e.g. Prove that "n is odd $\to n^2$ is odd"

$n = 2k+1$, $n^2 = (2k+1)^2 = 4k^2+4k+1 = 2(2k^2+2k)+1$

$k \in \mathbb{Z}$ ∴ $2k^2+k \in \mathbb{Z}$ ∴ $n^2 =$ odd

Def Proof by contraposition (one of indirect proofs)

∴ we wanna know $p \to q$ so instead prove $\neg q \to \neg p$

e.g. Prove that "if $3n+2$ is odd, then n is odd"

contraposition : if n is even, then $3n+2$ is even

$n = 2k$ where $k \in \mathbb{Z}$ ∴ $3n+2 = 6k+2 = 2(3k+1)$

Since $3k+1 \in \mathbb{Z}$, $3n+2 =$ even

Since contraposition is true, if $3n+2$ is odd, then n is odd

Jan 24, 2017

Def For the statement $p \to q$, if we can show p is false, then we have a proof, called a vacuous proof which can be a trivial proof

e.g. Let $P(n) =$ "$n > 1 \to n^2 > 1$", show $P(0)$ is true

$(0 > 1) \to (0^2 > 1)$ is true since $F \to F$ is T ← vacuous proof

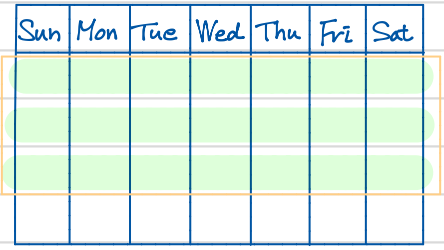
p.84

e.g. $P(n) =$ "$a, b \in \mathbb{Z}^+ \wedge a \ge b \to a^n \ge b^n$", show $P(0)$ is true

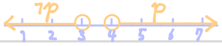
$P(0) =$ "$a . b \in \mathbb{Z}^+ \wedge a \ge b \to a^0 \ge b^0$" $a^0 = b^0 = 1$ ← trivial proof

Def **Proof by Contradiction** (one type of indirect proof) shows a statement p or $\neg p$ is false by using contradiction p.86

e.g. Show that at least 22 days must fall on the same day of the week



p (that you wanna show) = original statement



Let $\neg p$ = "at most three days of 22 days must fall on the same day of the week"

But we have only 7 days to chose from a week.

← Once we've chosen 21 days, every calender day has been picked at least 3 times. (There's no 8th day in a week)

Therefore, $\neg p$ is false (p is true) Q.E.D.

* Proof Methods & Strategy

- **Exhaustive Proof** ⋯ showing all examples p.92

  e.g. Shows that $2^n < 100$ if $n < 7$ $2^1 = 2 < 100$, $2^2 = 4 < 100$, $2^3 = 8 < 100$, $2^4 = 16 < 100$, $2^5 = 32 < 100$, $2^6 = 64 < 100$, $2^7 = 128 > 100$

- **Proof by Cases** : $(p_1 \lor p_2 \lor p_3 \lor p_4) \to q$

  e.g. Shows that $|xy| = |x||y|$ where $x, y \in \mathbb{R}$

  case(i) $x \geq 0$, $y \geq 0$ = $|xy| = xy = |x||y|$

  case(ii) $x \geq 0$, $y < 0$: $|xy| = -xy = |x||y|$

  case(iii) $x < 0$, $y \geq 0$: $|xy| = -xy = |x||y|$

  case(iv) $x < 0$, $y < 0$: $|xy| = xy = |x||y|$

  Therefore, in all possible cases, $|xy| = |x||y|$ ← exhaust all possibilities

- **Existence Proof** ⋯ shows $\exists x P(x)$ p.96

  [**Constructive** ⋯ shows an actual example of x s.t $P(x)$ is true

  [**Nonconstructive** ⋯ doesn't show an element x but shows its existense

  e.g. Prove the Thm : $\exists x, y \in (\mathbb{R} - \mathbb{Q})$ s.t. x^y is a rational number

  If $x = y = \sqrt{2}$, $x^y = \sqrt{2}^{\sqrt{2}}$

  If $\sqrt{2}^{\sqrt{2}}$ is rational, we've done ← constructive way

  nonconstructive way

  Otherwise, $\sqrt{2}^{\sqrt{2}}$ is irrational, then let $x = \sqrt{2}^{\sqrt{2}}$, $y = \sqrt{2}$ $\therefore x^y = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^2 = 2 \in \mathbb{R}$

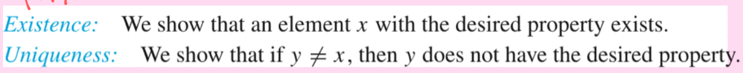
  Therefore, $\exists x, y \in (\mathbb{R} - \mathbb{Q})$ by showing 2 cases, but we don't know which case satisfies the statement

- **Uniqueness Proof** ⋯ shows if $y \neq x$, $P(y)$ is false where $P(x)$ is true p.99

  then $P(y) \to y = x$ (by contraposition)

e.g. $x, y \in \mathbb{R}$, $x > 0$, $y > 0$. Prove $\underbrace{(x+y)/2}_{\text{arthmatic mean}} > \underbrace{\sqrt{xy}}_{\text{metric mean}}$

P.99



Existence: We show that an element x with the desired property exists.

Uniqueness: We show that if $y \neq x$, then y does not have the desired property.

$(x+y)/2 > \sqrt{xy}$ $\Rightarrow (x+y)^2/4 = xy$ $\Leftrightarrow (x+y)^2 > 4xy$

$\therefore x^2 + 2xy + y^2 > 4xy$ $\Leftrightarrow x^2 - 2xy + y^2 > 0$ $\Leftrightarrow (x-y)^2 > 0$

$(x-y)^2 > 0$ where $x \neq y$ is true

So we conclude that if x and y are distinct positive real numbers, $(x+y)/2 > \sqrt{xy}$

- Backwards Reasoning (Proof strategy)

Jan 26, 2017

∴ Assume x and y are $\overset{x \neq y}{\text{distinct}}$ positive real numbers

* in this case, statements must be transformed with biconditional (↔)

- Looking for Counterexamples ∴ shows a statement false

p.102

① The conjection may be false

② Failing repeatedly to find counterexample sometimes give a hint to prove

③ Lack of counterexample is NOT proof

* Note = Prove or Disprove

e.g. Fermat's last theorem

p.106

$x^n + y^n = z^n$ has no solution in $x, y, z \in \mathbb{Z}$ with $xyz \neq 0$ whenever $n \in \mathbb{Z}$ with $n > 2$

Sets

Def = a set is unordered collection of all of numbers, elements, objects, things, and anything p.116

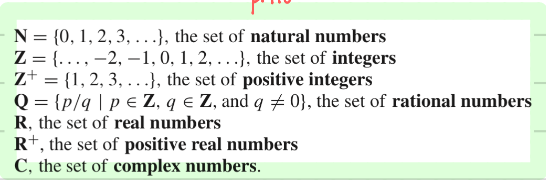
which is unlike a list (ordered collection)

e.g. vowels = $\{a, e, i, u, o\} = \{a, i, u, e, o\} = \{a, a, e, i, u, o, o, o\}$

$\mathbb{N} = \{1, 2, 3, \cdots\} = \{7, 14, \cdots\}$

$\mathbb{Z} = \{0, \pm 1, \pm 2 \cdots\}$

p.116



N = {0, 1, 2, 3, . . .}, the set of **natural numbers**
Z = {. . . , −2, −1, 0, 1, 2, . . .}, the set of **integers**
$\mathbf{Z}^+$ = {1, 2, 3, . . .}, the set of **positive integers**
Q = {p/q | $p \in \mathbf{Z}$, $q \in \mathbf{Z}$, and $q \neq 0$}, the set of **rational numbers**
R, the set of **real numbers**
$\mathbf{R}^+$, the set of **positive real numbers**
C, the set of **complex numbers**.

Def membership symbol $\in$ … $x \in S$ = "x is a member of S"

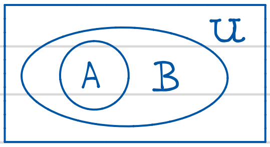
Def $A \subseteq B$ = "A is a subset of B" iff $\forall x \in A\ (x \in B)$ or iff $\forall x\ (x \in A \to x \in B)$

Def $A \subset B$ = "A is a proper subset of B" $\equiv (A \subseteq B \wedge A \neq B)$

Note: $A \subseteq B$ allows $A = B$

Def $\emptyset = \{\ \}$ = "null set", "empty set"

* Venn Diagrams



$\mathcal{U}$ = "universe"

Thm $\emptyset \subseteq S$ for any set S

need to show $\forall x (x \in \emptyset \to x \in S) \equiv \forall x (\underbrace{F \to x \in S}_{\text{vacuous proof p.84}}) \equiv \forall x\, T \equiv T$ Q.E.D

Def a power set of a given set S is the set of all subsets of S = $\mathcal{P}(S)$ p.121

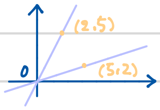
e.g. power set of the set $\{0, 1, 2\}$

$\mathcal{P}(\{0, 1, 2\}) = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}\}$

e.g. $\mathcal{P}(\emptyset) = \{\emptyset\}$, $\mathcal{P}(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$

* Ordered n-tuple … has n elements and order is important ↔ "two lists are equal to each other only if the same elements in the same order

e.g. if n=2, "ordered pairs" if n=3, "ordered triples"

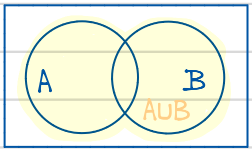


$(x, y) \neq (y, x)$

Def A and B are 2 sets, The Cartesian Product of A and B is $A \times B = \underset{\text{set of all}}{\{} (a \underset{\text{a comma b}}{,} b) \underset{\text{such that}}{|} a \in A \underset{\text{and}}{\wedge} b \in B \}$

$A \times B \times C = \{(a, b, c) \mid a \in A \wedge b \in B \wedge c \in C\}$

* Set Operations



Def "union of A and B" = $A \cup B = \{x \mid x \in A \vee x \in B\}$

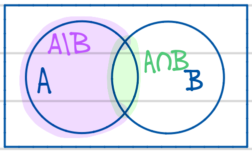
Def "intersection of A and B" = $A \cap B$ = "$\{x \mid x \in A \wedge x \in B\}$

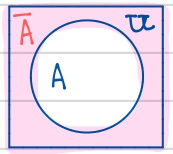
Def A and B are disjoint if $A \cap B = \emptyset$

Def set subtraction = $A - B$ (or $\underset{\text{backslash}}{A \setminus B}$) = $\{x \mid x \in A \wedge x \notin B\}$

if $A \subset B$,
$A - B = \emptyset$

Def Given a universe $\mathcal{U}$, $\mathcal{U} \setminus A$ also denoted $\overline{A}$, whic is a complement of A





p. 130

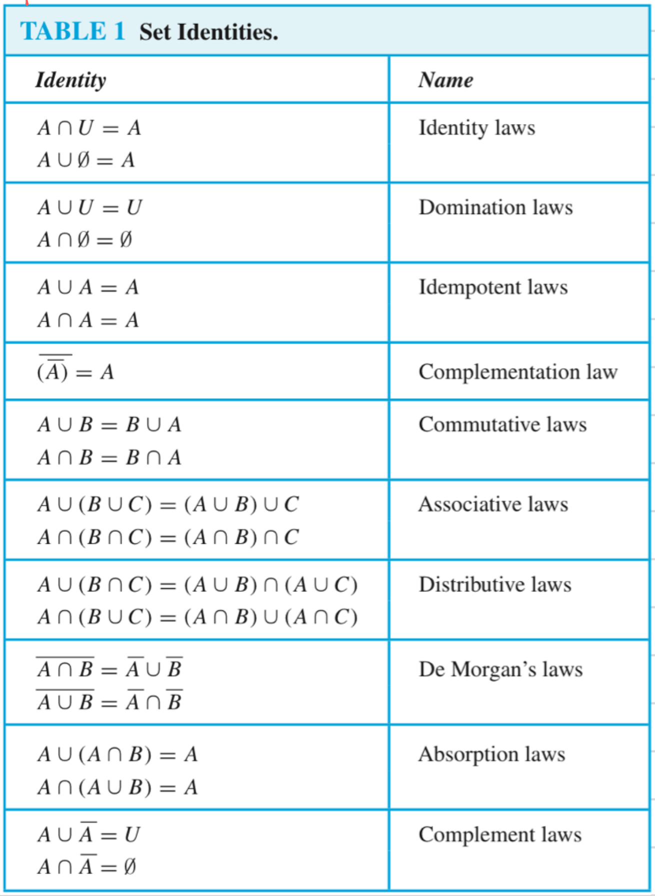


TABLE 1 **Set Identities.**

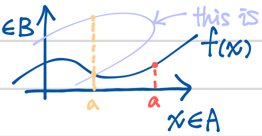
Identity	*Name*
$A \cap U = A$ $A \cup \emptyset = A$	Identity laws
$A \cup U = U$ $A \cap \emptyset = \emptyset$	Domination laws
$A \cup A = A$ $A \cap A = A$	Idempotent laws
$\overline{(\overline{A})} = A$	Complementation law
$A \cup B = B \cup A$ $A \cap B = B \cap A$	Commutative laws
$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$	Associative laws
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	Distributive laws
$\overline{A \cap B} = \overline{A} \cup \overline{B}$ $\overline{A \cup B} = \overline{A} \cap \overline{B}$	De Morgan's laws
$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$	Absorption laws
$A \cup \overline{A} = U$ $A \cap \overline{A} = \emptyset$	Complement laws

Functions

p.138

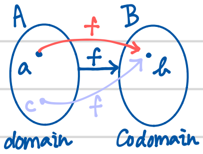
Def: a function (sometimes called map transformation) f from A to B takes every element of A to exactly one element of B

e.g. $y=f(x)$ $f: A \to B$



$f(a)=b$

*Note that $b \in B$ can result from multiple values of $a \in A$
but $f(a)$ has only one value



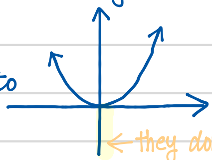
Jan 31, 2017

Def: b is the image of a under f
a is the pre-image of b under f

Def: Range of A under f $= \{b \in B \mid \exists a \in A\ f(a)=b\}$ ← range $\subseteq$ Co-domain

Def: function f is 1-to-1 (injective, an injection) iff $f(a)=f(b) \to a=b$
i.e. each b has only 1 pre-image
$\forall a \forall b\, (f(a)=f(b) \to a=b)$ or $\forall a \forall b\, (a \neq b \to f(a) \neq f(b))$

Def: function f is onto (surjective, a surjection) iff $b \in B\ \exists a \in\ f(a)=b$
i.e. every member of co-domain B is "covered" by the image of something in A
$\Leftrightarrow$ co-domain = range, $f(A)=B$
e.g. $A=\mathbb{R}$, $B=\mathbb{R}$ then $f(x)=x^2$ $(f: A \to B)$ is NOT onto
$A=\mathbb{R}$, $B=\mathbb{R}^+$ then $f(x)=x^2$ $(f: A \to B)$ is onto



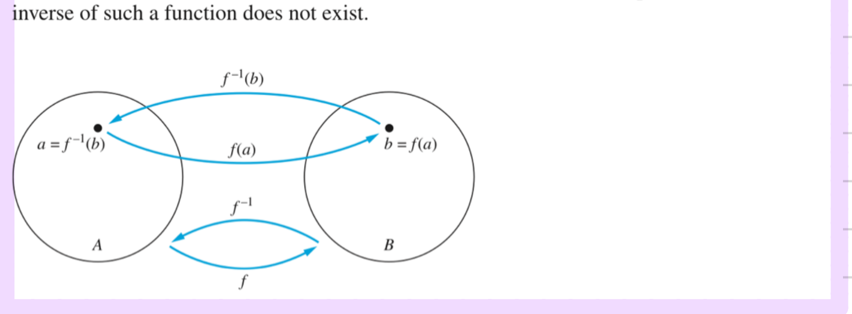
← they don't have pre-image where $B=\mathbb{R}$

Def: function f is bijective iff it's both injective and surjective aka "1-to-1 correspondence" P.141

Def: Let $f: A \to B$ be bijective. That means $\forall b \in B\ \exists a \in A\ f(a)$
The inverse of f, $f^{-1}(b)=a$, to be the a s.t. $f(a)=b$ i.e. $f^{-1}(b)=a$ if f is bijective function

*If f is not bijective, f^{-1} is not defined p.145

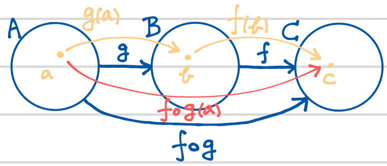
Thm $f(f^{-1}(b))=b$ and $f^{-1}(f(a))=a$



A one-to-one correspondence is called **invertible** because we can define an inverse of this function. A function is **not invertible** if it is not a one-to-one correspondence, because the inverse of such a function does not exist.

Def: Composition of function
$(f \circ f^{-1})(b) = f(f^{-1}(b)) = b$
$(f^{-1} \circ f)(a) = f^{-1}(f(a)) = a$

where $g: A \to B$, $f: B \to C$ (also $a \in A$, $b \in B$)
$(f \circ g)(a) = f(g(a)) = f(b) = c \in C$



Def: The graph of $f: A \to B$ is $\{(a,b) \mid a \in A \cap f(a)=b\}$ p.148
↳ It doesn't need to show visible images on xy-coordinate ⇒ Graph is a set of pair (n-tuple)

Def: f is well-defined if $\forall x \in D \ \exists y \in C$ s.t. $y = f(x)$ p.149

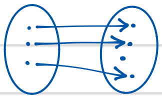
e.g $\text{floor}(x) = \lfloor x \rfloor \equiv \overset{\max}{y \in \mathbb{Z}} = (y \le x)$

$\lfloor \pi \rfloor = 3$, $\lfloor -4 \rfloor = -4$

$\text{ceiling}(x) = \lceil x \rceil \equiv \overset{\min}{y \in \mathbb{Z}} = (y \ge x)$

$\lceil \pi \rceil = 4$, $\lceil -\pi \rceil = -3$

* Inverse f^{cn}



This f^{cn} has No inverse since $\nexists x \in A \ f(x) = y \in \mathbb{Z}$

Relations

Def: Let A, B be sets, recall that $A \times B = \{(a, b) \mid a \in A, b \in B\}$

A relation R is some subset of $A \times B$

We write aRb to mean "a is related to b under R"

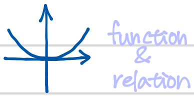
e.g. $A = \{$ students of UCI $\}$

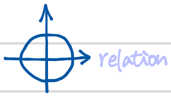
$B = \{$ classes of UCI $\}$

Let $s \in A$, $c \in B$, and $sRc =$ "student s is taking class c"

* In general, relation is many-to-many (NOT 1-to-1 or onto)

* function $\subset$ Relation





*some formulas

aRb

$a = b$

$a \le b$

$a \ge b$ etc...

e.g, relation on $\mathbb{Z}^+$ is $R = \mid$ = "is divisible by"

$\therefore \{(4,2), (6,3), (9.3)\} \subset R$

$(4.3) \not\subset R$

also, $\nexists k > 1 \in \mathbb{Z}^+$ s.t. $(11, k) \in R \iff$ "11 is a prime number"

*Binary Matrix ... 1 is true (in relation) P.181

$$\begin{array}{c|ccccc|c} & a & b & c & d & e & \in B \\ \hline 1 & 0 & 0 & 1 & 0 & 1 & \\ 2 & 0 & 1 & 1 & 0 & 1 & \\ 3 & 1 & 0 & 1 & 1 & 1 & \\ \hline \in A & & & & & & \end{array}$$

e.g. $1Rc$, $1Re$, $3Rd$

$2\cancel{R}a$

* How many possible relations exist?

(Recall $R \le \underset{\text{maximum}}{A \times B}$ $|A| = n$, $|B| = m$, each entry is 0 or 1

total # of possible relations is $\boxed{2^{\boxed{n \times m}}}$ — entry, ↳options

e.g, on 3x5 matrix, there's

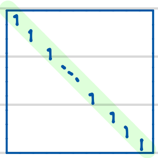
$2^{3\times5} = 2^{15} = 32768$ possible relations

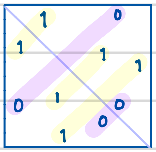
* Properties of Relations on A×A

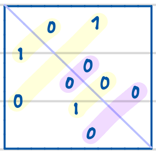
① $(a,a) \in R \equiv$ "A is reflexive" p.576

② $[(a,b) \in R \leftrightarrow (b,a) \in R] \equiv$ "A is symmetric" p.577

③ $\forall a,b \in A\, [(a,b) \in R \wedge (b,a) = R \rightarrow a = b] \equiv [\nexists (a.b) \in R\, ((b,a) \in R \wedge (a \neq b)] \equiv$ "R is anti-symmetric"

①


②


③


no symmetric pair of relation (where element is 1)

④ $[(a.b) \in R \wedge (b,c) \in R \rightarrow (a.c) \in R] \equiv$ "R is transitive" p.578

e.g. $=, >, <$

But $\neq$ is Not transitive

* Combining Relations p579

Note: Relation is just a set of ordered pairs ⇒ using set operations on relations to define new relations

e.g. A = { students }, B = { courses }

R_1 = { has taken }, R_2 = { need to take in order to graduate"

What do they mean $R_1 \cap R_2$, $R_1 \cup R_2$, $R_1 \oplus R_2$, $R_1 - R_2$, and $R_2 - R_1$?

$R_1 \cap R_2$ = "all courses a student needs to taken and has already taken"

$R_1 \cup R_2$ = "all courses a student needs to taken $\overset{\text{OR}/\vee}{+}$ has already taken"

$R_1 \oplus R_2$ = "all elective courses that a student has already taken + required courses to graduate but not taken yet"

$R_1 - R_2$ = " all elective courses that a student has already taken"

$R_2 - R_1$ = " all required courses to graduate but not taken yet"

* Composition of Relations p.580

Let $R \subseteq A \times B$, $S \subseteq B \times C$

then $S \circ R = \{ (a.c) \mid a \in A, c \in C, \exists b \in B\, (a.b) \in R \wedge (b,c) \in S \}$

* There can be multiple cs for any a, and vice versa

e.g. R = "is the parent of"

$(a,b) \in R$, $(b,c) \in R$ $(a.c) \in R \circ R$ ← a has more than 1 grandparents

* Recursion relations p.580

$R^1 = R$, $R^2 = R \circ R$, ..., $R^{n+1} = R^n \circ R$

<u>Thm</u> R on a set A is transitive iff $R^n \subseteq R$ $\forall n > 0$

(→) suppose $R^n \subseteq R$ $\forall > 0$, $R^2 \subseteq R$ is true

Note that $(a,b) \in R$ and $(b,c) \in R \rightarrow (a,c) \in R \circ R = R^2$ since $R^2 \subseteq R$, and $(a,c) \in R$

(←) it's too hard to prove now...

* n-ary Relations

''' defines relationship between multiple entries simultaneously

Def= given sets $A_1, A_2, \cdots, A_n$ (Domains), n is the degree of a relation $R \subseteq A_1 \times A_2 \times \cdots \times A_n$

e.g. $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ s.t. $a < b < c$

e.g. (A, F, S, D, T) = ("Airline," "flight #", "departure city", "distination", "departure time") p.583

e.g. p.589

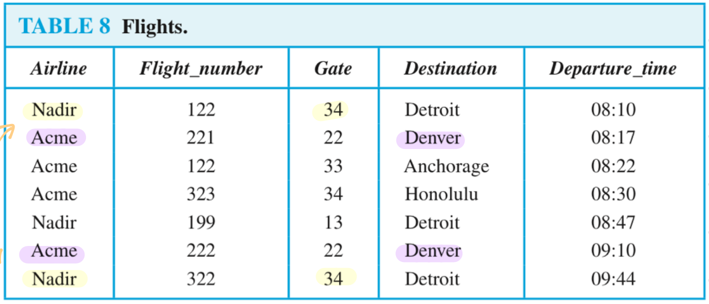
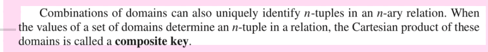


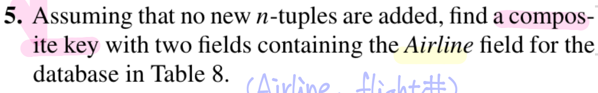
TABLE 8 **Flights.**

Airline	*Flight_number*	*Gate*	*Destination*	*Departure_time*
Nadir	122	34	Detroit	08:10
Acme	221	22	Denver	08:17
Acme	122	33	Anchorage	08:22
Acme	323	34	Honolulu	08:30
Nadir	199	13	Detroit	08:47
Acme	222	22	Denver	09:10
Nadir	322	34	Detroit	09:44

NOT unique



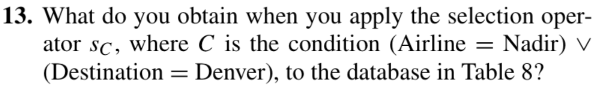
Combinations of domains can also uniquely identify n-tuples in an n-ary relation. When the values of a set of domains determine an n-tuple in a relation, the Cartesian product of these domains is called a **composite key**.



5. Assuming that no new n-tuples are added, find a composite key with two fields containing the *Airline* field for the database in Table 8.

(Airline, flight #),

(Airline, departure time)

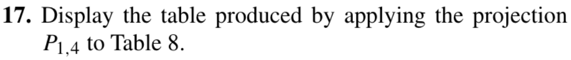


13. What do you obtain when you apply the selection operator s_C, where C is the condition (Airline = Nadir) ∨ (Destination = Denver), to the database in Table 8?

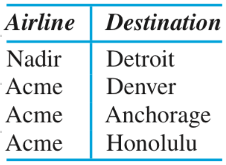
(Nadir, 122, 34, Detroit, 08:10), (Nadir, 199, 13, Detroit, 08:47)

(Nadir, 322, 34, Detroit, 09:44), (Acme, 221, 22, Denver, 08:17),

(Acme, 222, 22, Denver, 09:10)

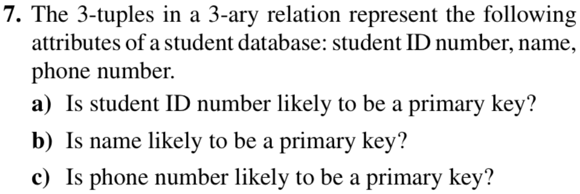


17. Display the table produced by applying the projection $P_{1,4}$ to Table 8.



Airline	*Destination*
Nadir	Detroit
Acme	Denver
Acme	Anchorage
Acme	Honolulu

e.g.



7. The 3-tuples in a 3-ary relation represent the following attributes of a student database: student ID number, name, phone number.

a) Is student ID number likely to be a primary key?

b) Is name likely to be a primary key?

c) Is phone number likely to be a primary key?

a) Yes (# of key = degree n)

b) No

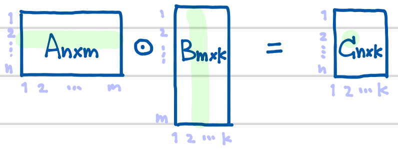
c) No

* Brief Review of Matrix

Feb 7, 2017

$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$ 2 rows, 3 clolumns "2x3"

- Addition : element by element (both must be the same size)
- Multiplication (Dot Product)



Let A be an nxm matrix and B be an mxk matrix. The product of A and B denoted by AB is the nxk matrix and its element c_{ij} $(1 \le i \le n,\ 1 \le j \le k)$ is p.179

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{im}b_{mj}$$

$$= \sum_{x=1}^{m} \underbrace{a_{ix}}_{\text{i-th row in A}} \underbrace{b_{xj}}_{\text{j-th column in B}}$$

* Amount of computation

each element of C costs m scalar multiplication & (m-1) additions ($\sum_{x=1}^{m} a_{ix}b_{xj}$)

total # of elements in C is nxk

∴ Total cost is nxmxk

Thm (AB)C = A(BC)

But the amount of computation could be different

e.g. Let $A_{10\times20}$, $B_{20\times30}$, $C_{30\times40}$

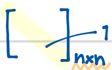
cost of $(AB)C = (\underbrace{10\times20\times30}_{}) + (\underbrace{10\times30\times40}_{}) = 6000 + 12000 = 18000$

AB (AB)C since $(AB)_{10\times30}$

" $A(BC) = (20\times30\times40) + (10\times20\times40) = 24000 + 8000 = 32000$

much more expensive ⇒ From CS prospect, we choose (AB)C in this case (faster)

- Property of Matrix

I = identity =


A^T = transpose = "flip across diagonal"

if $A^T = A$, A is a symmetric matrix ← if $A^T = A$, A is a symmetric

* Representing Relations (binary relations)

- list of ordered pairs $R \subseteq A \times B = \{(a_1, b_1), (a_1, b_2) \cdots (a_x, b_y)\}$
- Matrix (Boolean)

$$\begin{array}{c|cccccc} & & & \xrightarrow{\quad\quad} B & & & \\ \hline A \downarrow & 0 & 1 & 0 & 1 & 0 & 1 \\ & 1 & 0 & 1 & 1 & 0 & 1 \\ & & & \vdots & & & \end{array}$$

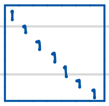
where $M_R = [m_{ij}] = \begin{cases} 1 & \text{if } (a_i, b_j) \in R \\ 0 & \text{otherwise} \end{cases}$

e.g. M_R where R = "i devids j" i, j < 5

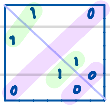
$$\begin{array}{c|c|c|c|c|} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 2 & 0 & 1 & 0 & 1 \\ \hline 3 & 0 & 0 & 1 & 0 \\ \hline 4 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

← antisymmetric

R is reflexive iff aRa ∀a∈A

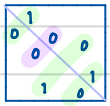


Symmetric iff aRb ↔ bRa



anti-symmetric iff aRb ∧ bRa → a=b

↑ the element has 1



I is { reflexive
symmetric
anti-symmetric

Composition of Relations p.182

say R relates A to B, S relates B to C

Define M_R and M_S as above, then $M_{R\otimes S}$ relates A to C s.t.

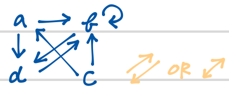
$M_{R\otimes S} = [t_{ij}]$, $t_{ij} = 1$ iff $\exists k\,(a_i R b_k \wedge b_k R c_j)$

Then $M_{R\otimes S}$ = Boolean Product of matricies $M_R \odot M_S$

Matrix multiplication with × being ∧, + being ∨

* Digraphs (Directed Graphs)

e.g.



aRb, aRd, bRd, dRb, cRb, cRa, bRb self-loop

arrows = "edge" , letters = "node" in the graph

Note: if R is symmetric, all arrows must go both ways ⇒ the graph is "undirected"

M_R

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 1 & 0 & 1 \\ c & 1 & 1 & 0 & 0 \\ d & 0 & 1 & 0 & 0 \end{array}$$

output

input

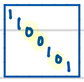
* Closure of Relations p.597

Given R under property P, closure of R under P is the smallest new relation S that both

① has property P

② contains R as a subset

e.g M_R =



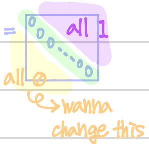
reflexive clausure ∴ add 3 elements

e.g Reflexive closure of $R = \{(a,b) \in \mathbb{Z} \mid a < b\}$

$S = R \cup \overset{P}{\Delta} = \{(a,b) \mid a<b\} \cup \{(a,a) \mid a \in \mathbb{Z}\} = \{(a,b) \mid a \leq b\}$ ∴ reflexive enclosure of < is ≤

e.g. Let R be "<" on integers. Create a symmetric closure of R

M_R =



wanna add minimum Relation

while keeping the original relation R

∴ closure (R) = $R \cup \{(a,b) \mid (b,a) \in R\} = \{(a,b) \mid a<b \vee a>b\} = \{(a,b) \mid a \neq b\}$

symmetric closure of "<" is "<, >" AKA "≠"

tech Def closure = Let R be a relation on A×A, Let P be any property of relations (e.g, reflexive, symmetric, transitive etc)

If ∃ a relation S s.t. S is a subset of every relation satisfying P that contains R,

then S is the closure of R under P

i.e, $\exists S\,(P(S) \wedge \forall T\,(R \subset T \wedge P(T) \rightarrow S \subseteq T))$

⇒ if this evaluate is TRUE, S is the enclosure

* Transitive Closure ← The hardest one

Feb 14, 2017

Let $R = \{(1,3), (1,4), (2,1), (3,2)\}$ Note: R is transitive if $aRb \wedge bRc \to aRc$

Step Ⅰ since ${}_1R_3$, ${}_3R_2$: is ${}_1R_2$? → No then add it ⇒ now $(1,2) \in R$

what else? ${}_2R_1$, ${}_1R_3 \to (2,3)$
${}_2R_1$, ${}_1R_4 \to (2,4)$ } added
${}_3R_2$, ${}_2R_1 \to (3,1)$

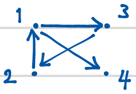
Step Ⅱ Now, need to think the added relations too

${}_1R_2$, ${}_2R_1 \to (1,1)$
${}_3R_2$, ${}_2R_{\substack{3\\4}} \to (3,3), (3,4)$
${}_2R_3$, ${}_3R_2 \to (2,2)$

step Ⅲ Again, think about the new relations

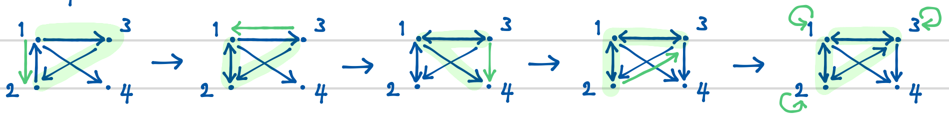
${}_1R_3$, ${}_3R_{\substack{3\\4}} \to (1,3), (1,4)$ ⇒ Transitive closure of R is all these pairs with R

Let's consider an easier way



recall def of transitive

if $(a \xrightarrow{R} b) \wedge (b \xrightarrow{R} c) \to (a \xrightarrow{R} c)$ ← Considering Pathes here



Path on directed edge

Def = a graph is a set of nodes and a set of ordered pair on nodes called edges

Def = $\overset{xRy}{(x,y)}$ is a directed edge.

Path P is a sequence of edges $e_i = (x_i, y_i)$ s.t. $y_i = x_{i+1}$ (second node in e_i is the first node in e_{i+1})

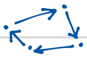
e.g. $(a,b), (b,c), (c,d), (d,c) \cdots$

Def = Path length is the number of edges in the graph (= # node − 1)

e.g. $a \to b \to c$ path length = 2, 3 nodes

Note ① Path from a node to itself can be length zero if no self-loop or any non-negative integer if aRa ←?

② If k edges & last node = the first node where $k>0$. this is called a circuit or circle



③ both edge and nodes can appear more than once

Recall composition of relations $R \circ R = R^2$

In graph terms, R^2 = set of path of length 2

e.g. if $aRb \wedge bRc$, $\overset{path}{(a,c)}$ is in R^2

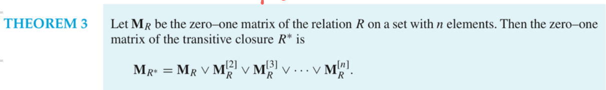
$\underbrace{R \circ R \circ \cdots \circ R}_{k \in \mathbb{N}} = R^k$ = set of path of length k Note: by definition, $R^0 \equiv \mathbb{I}$ = path of length 0

Def: $R^* = \bigcup_{k=1}^{\infty} R^k$ = "connectivity relation on R" = "reachibility of graph on R"

≡ transitive closure of R

Thm $R^* = \bigcup_{k=1}^{n} R^k$ where n = # nodes (elements of A)

P. 602



THEOREM 3 Let $\mathbf{M}_R$ be the zero–one matrix of the relation R on a set with n elements. Then the zero–one matrix of the transitive closure R^* is

$$\mathbf{M}_{R^*} = \mathbf{M}_R \vee \mathbf{M}_R^{[2]} \vee \mathbf{M}_R^{[3]} \vee \cdots \vee \mathbf{M}_R^{[n]}.$$

* cost of computing transitive enclosure of a relation R on A×A where $\underbrace{|A|}_{\text{size of A}}=n$

R can be represented as a binary matrix n×n = M

R^2 can be computed as M×M which costs $\approx n^3$, n times → total cost is at most $O(n^4)$

… actually can do R^* in n^3 times

* Warshall's Algorithm — think of connectivity

Let $\underbrace{W_0}_{\text{Matrix}} = M_R$ = matrix representing R (directed graph)

Def = W_k ≡ Matrix of reachability (i, j) but only allowed to use intermediate nodes 1, ⋯, k

P.606

LEMMA 2 Let $\mathbf{W}_k = [w_{ij}^{[k]}]$ be the zero–one matrix that has a 1 in its (i, j)th position if and only if there is a path from v_i to v_j with interior vertices from the set $\{v_1, v_2, \ldots, v_k\}$. Then

$$w_{ij}^{[k]} = w_{ij}^{[k-1]} \vee (w_{ik}^{[k-1]} \wedge w_{kj}^{[k-1]}),$$

whenever i, j, and k are positive integers not exceeding n.

e.g. (P. 605)

EXAMPLE 8 Let R be the relation with directed graph shown in Figure 3. Let a, b, c, d be a listing of the elements of the set. Find the matrices $\mathbf{W}_0$, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$. The matrix $\mathbf{W}_4$ is the transitive closure of R.

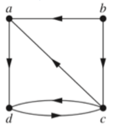


FIGURE 3 **The Directed Graph of the Relation *R*.**

Solution: Let $v_1 = a$, $v_2 = b$, $v_3 = c$, and $v_4 = d$. $\mathbf{W}_0$ is the matrix of the relation. Hence,

$$\mathbf{W}_0 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

$\mathbf{W}_1$ has 1 as its (i, j)th entry if there is a path from v_i to v_j that has only $v_1 = a$ as an interior vertex. Note that all paths of length one can still be used because they have no interior vertices. Also, there is now an allowable path from b to d, namely, b, a, d. Hence,

$$\mathbf{W}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

← now there is a path from b to d via a

$\mathbf{W}_2$ has 1 as its (i, j)th entry if there is a path from v_i to v_j that has only $v_1 = a$ and/or $v_2 = b$ as its interior vertices, if any. Because there are no edges that have b as a terminal vertex, no new paths are obtained when we permit b to be an interior vertex. Hence, $\mathbf{W}_2 = \mathbf{W}_1$.

$\mathbf{W}_3$ has 1 as its (i, j)th entry if there is a path from v_i to v_j that has only $v_1 = a$, $v_2 = b$, and/or $v_3 = c$ as its interior vertices, if any. We now have paths from d to a, namely, d, c, a, and from d to d, namely, d, c, d. Hence,

$$\mathbf{W}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

↖ now there is a path from d to a via c

Finally, $\mathbf{W}_4$ has 1 as its (i, j)th entry if there is a path from v_i to v_j that has $v_1 = a$, $v_2 = b$, $v_3 = c$, and/or $v_4 = d$ as interior vertices, if any. Because these are all the vertices of the graph, this entry is 1 if and only if there is a path from v_i to v_j. Hence,

$$\mathbf{W}_4 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

This last matrix, $\mathbf{W}_4$, is the matrix of the transitive closure. ◀

* Equivalence Relations

Def: a relation R on set A is called an Equivalence Relation (ER) if it is reflexive, symmetric, and transitive.

e.g. old-style C-language only used first 8 charactors of carriable name to identify

`int thisVariable;`
`int thisVariation;` } equivalent names in old C-language

Let R be a relation on all strings of all length where aRb if they share first 8 characters

reflexive ✓, symmetric ✓, transitive ✓

∴ given an ER, a set of strings starting with "thisVari" are called "Equivalent class"

e.g. **EXAMPLE 9** What are the equivalence classes of 0 and 1 for congruence modulo 4? (p.610)

$\{0, 4, 8, 16 \cdots\} = [0]$ = "equivalence class of 0"

$\{1, 5, 9, 17 \cdots\} = [1]$ = " " of 1"

$\{2, 6, 10, 18 \cdots\} = [2]$ = " " of 2"

$\{3, 7, 11, 19 \cdots\} = [3]$ = " " of 3"

all reflexive, symmetric, and transitive

Feb 16, 2017

Def: Two related by an E.R, they are called equivalent, $a \sim b$

speech 25m

Note: "$a \sim b$" order is NOT important (aRb is symmetric here)

e.g. is $|x-y|<1$ on E.R?

reflexive? $|x-x| = 0 < 1$ ✓ ; symmetric? $|x-y| = |y-x|$ ✓ ; transitive? $|x-y| < 1 \wedge |y-z| < 1 \not\rightarrow |x-z| < 1$ X

$\Rightarrow$ NOT E.R.

Def: given $a \in A$ and on E.R. let $[a]_R = \{b \in A \mid aRb\}$ called "equivalence class of"

we say a is a "representation" of $[a]_R$ but any number of $[a]_R$ would suffice

a can be b since E.R → reflexive

e.g. what is $[3]_R$ if $R = \{(a, b) \in \mathbb{Z}^+ \mid a = b \pmod 4\}$

$[3]_R = \{3, 7, 11, 15, 19 \cdots\} = [19]_R$

e.g. **26.** What are the equivalence classes of the equivalence relations in Exercise 1? HW p.616

a) $\{(0, 0), (1, 1), (2, 2), (3, 3)\}$

~~**b)** $\{(0, 0), (0, 2), (2, 0), (2, 2), (2, 3), (3, 2), (3, 3)\}$~~

c) $\{(0, 0), (1, 1), (1, 2), (2, 1), (2, 2), (3, 3)\}$

~~**d)** $\{(0, 0), (1, 1), (1, 3), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\}$~~

~~**e)** $\{(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 2), (3, 3)\}$~~

a) $[0]_R = \{0\}$, $[1]_R = \{1\}$, $[2]_R = \{2\}$, $[3]_R = \{3\}$

c) $[0]_R = \{0\}$, $[1]_R = \{1, 2\}$, $([2]_R = \{1, 2\} = [1]_R)$, $[3]_R = \{3\}$

Thm $aRb \leftrightarrow [a]_R = [b]_R \longleftrightarrow [a]_R \cap [b]_R \neq \emptyset$ on E.R.

corollary: $a \cancel{R} b \leftrightarrow [a]_R \cap [b]_R = \emptyset$

corollary: $\bigcup_{a \in A} [a]_R = A$ ← since A is on E.R, reflexive

union of all elements

p.613

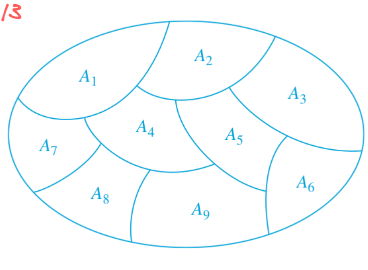


FIGURE 1 **A Partition of a Set.**

partition diagram

→ stronger Venn diagram

More generally, given only set of subsets $A_i \subseteq A$, we say that $\{A_i\}$ form a partition of A iff $\forall i \{A_i\} \neq \emptyset$, $i \neq j \Rightarrow A_i \cap A_j = \emptyset$, and $\underset{\text{for all } i}{\bigcup A_i} = A$

A partition of A implicitly defines a relation R on A

Thm Let R be an E.R, on A, then the equivalence classes of R form a partition of A

Conversely, given a partition $[A_i]$ of A, $\exists$ E.R. R that has A_i as its equivalence classes

* Partial orderings p.618

Def= given a relation R, R is called a partial ordering of A if R is reflexive, anti-symmetric, and transitive on A AKA "posets"

e.g, $\geq$ on $\mathbb{Z}$

$a \geq a$ ∵ reflexive, $a \geq b \wedge a \leq b \rightarrow a = b$ ∵ anti-symmetric, $a \geq b \wedge b \geq c \rightarrow a \geq c$ ∵ transitive

⇒ $\geq$ on $\mathbb{Z}$ is parcial ordering on $\mathbb{Z}$

e.g. devides "|" on $\mathbb{Z}^+$

$a|a$ ∵ relfexive, $a|b \wedge b|a \rightarrow a = b$ ∵ anti-symmetric, $a|b \wedge b|c \rightarrow a|c$ ∵ transitive

⇒ "|" on $\mathbb{Z}^+$ is a parcial ordering on $\mathbb{Z}^+$

Def= $aRb \vee bRa$ we say a and b are "comparable"

Def= If all pairs in A are comparable under R,

then R is a total ordering

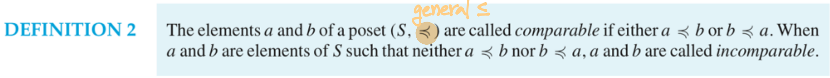
∴ total ordering ⊆ partial ordering

*why called "parcial" ordering? (p.619)

in the e.g of "|", 3 and 9 are comparable ∵ $3|9$

but 5 and 7 are not comparable (∵ $5 \nmid 7$ and $7 \nmid 5$)

→ part of $\mathbb{Z}^+$ are comparable



DEFINITION 2 The elements *a* and *b* of a poset $(S, \preccurlyeq)$ are called *comparable* if either $a \preccurlyeq b$ or $b \preccurlyeq a$. When *a* and *b* are elements of *S* such that neither $a \preccurlyeq b$ nor $b \preccurlyeq a$, *a* and *b* are called *incomparable*.

general ≤

Def: If R on A is a total ordering,

and every non-empty set of A has a least element, then A is well-ordered under R

e.g, $\mathbb{Z}$ is NOT well-ordered since there's no least element

e.g, the lexicographic ordering is well-ordered set

↳ $(a_1, a_2) \prec (b_1, b_2)$ if $a_1 \prec b_1 \vee (a_1 = b_1 \wedge a_2 \prec b_2)$

Feb 21, 2017

e.g, words in a dictionary : shorter words come first before longer words if the shorter word is the short of the longer word

e.g, "and" < "andromeda"

* Hasse diagram on posets p.622

step 1 create directed acyclic graph (DAGs) → check ICS46 Note (Algorithm)

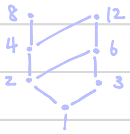
step 2 remove all edges that can be inferred from other edges → reflexive, transitive

e.g, $\geq$ on $\{1, 2, 3, 4\}$

original directed graph

*going up (input is bottom)

e.g, "|" divids on $A = \{1, 2, 3, 4, 6, 8, 12\}$



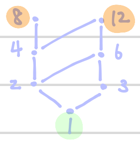
if Hass-diagram is written well,

max & min are obvious

Def= a maximal element has no dements grater (≺) than itself

a minimal element has no dements smaller (≻) than itself

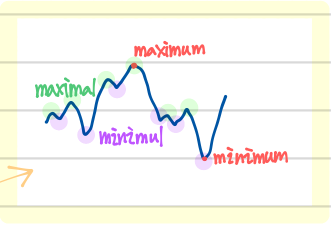
e.g, from above, 1 is minimal & 8 and 12 are maximal



Def= an element $a \in A$ is maximum (greatest) if $b \leq a \quad \forall b \in A$

minimum (least) if $a \leq b \quad \forall b \in A$

e.g, from above, 1 is minimum, no maximum ← greatest/ least are unique if they exists

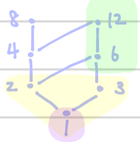


Def: given set S, and subset $A \subseteq S$, $u \in S$

u is called "upper bound on A" if $a \preceq u, \forall a \in A$ (Note: u must be comparable to all elements in A)

"lower bound on A" if $a \succeq u, \forall a \in A$ (Note: u must be comparable to all elements in A)

e.g, from previous page ("1"), if $A = \{1, 2, 3\}$



6 and 12 are both upper bound. (4 & 8 are NOT since it's not comparable with $3 \in A$)

1 is a lower bound on any $A \subseteq S$

Def: the Least upper bound is the smallest of all upper bounds (LUB)

the Gratest lower bound is the largest of all lower bounds (GLB)

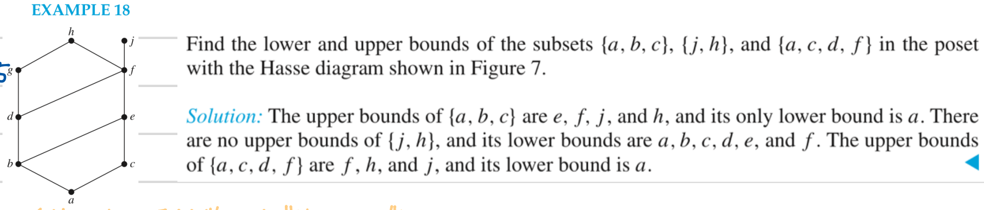
e.g, from previous page ("1") if $A = \{6, 12\}$

lower bound is 1, 2, 3 → gratest one is 3 (GLB)

upper bound is 12 → least one is 12 (LUB)

e.g,

p.625



EXAMPLE 18

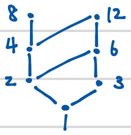
Find the lower and upper bounds of the subsets $\{a, b, c\}$, $\{j, h\}$, and $\{a, c, d, f\}$ in the poset with the Hasse diagram shown in Figure 7.

Solution: The upper bounds of $\{a, b, c\}$ are e, f, j, and h, and its only lower bound is a. There are no upper bounds of $\{j, h\}$, and its lower bounds are a, b, c, d, e, and f. The upper bounds of $\{a, c, d, f\}$ are f, h, and j, and its lower bound is a. ◀

* Topological Sort (Also check ICS 46 Note "Algorithm I")

Basically lists the elements bottom-up in Hasse diagram s.t. only comparable items matter in the ordering. incomparable items can be shuffled.

e.g, previous page

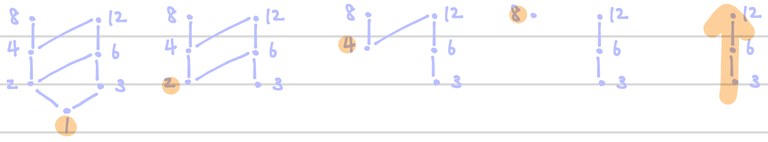


1, 2, 3, 4, 6, 8, 12
1, 3, 2, 6, 4, 12, 8
} example of few valid topological sort

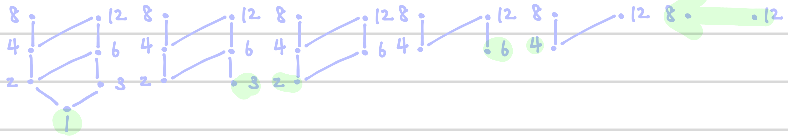
* Lemma : every finite non-empty poset has at least one minimal element

* Algorythm

```
k=1
while S ≠ ∅
    let a_k = any minimal element of S
    let S = S - {a_k}
    let k = k+1
end while
return (a_1, a_2, ..., a_k)
```



k	1	2	3	4	5	6	7
minimals (circle the chosen one)	1	2, 3	4, 3	8, 3	3	6	12
	1	2, 3	2	6, 4	4	8, 12	8



Feb 23, 2017

* strict ordering

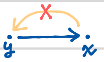
Def: say $\preceq$ is a partial ordering, then the associated strict ordering $\prec$ removes the reflexive ordering

e.g, $\leq \rightarrow <$ on numbers. $\subseteq \rightarrow \subset$ on sets

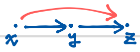
Recall Directed Acycle Graph (DAGs): DAGs represent strict orderings because (assume no self-loop)

because ① no reflexive elements (irreflexive)

② anti-symmetric $y < x \rightarrow x \nless y$



③ transitive $x < y \wedge y < z \rightarrow x < z$



implied by transitive closure

Boolean Algebra ☺ p.811

* different notations (duality) for $T, F, \wedge, \vee, \neg x$, ⋯ etc
$1, 0, \cdot, +, \overline{x}$

practice $= (T \wedge F) \vee \neg(F \vee T) = F \vee \neg T = F \vee F = F$
$1 \cdot 0 + \overline{(0+1)} = 0 + \overline{1} = 0 + 0 = 0$ } same

e.g. $x + y = 1 \Leftrightarrow x = 1 \vee y = 1$

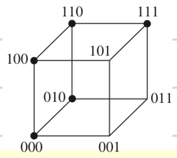
$1+1=1,\ 1+0=1,\ 0+1=1,\ 0+0=0$

$x \cdot y = 1$ iff $x = 1 \wedge y = 1$

$\neg x \equiv \overline{x}$

The function $F(x, y, z) = xy + \overline{z}$ from B^3 to B from Example 5 can be represented by distinguishing the vertices that correspond to the five 3-tuples (1, 1, 1), (1, 1, 0), (1, 0, 0), (0, 1, 0), and (0, 0, 0), where $F(x, y, z) = 1$, as shown in Figure 1. These vertices are displayed using solid black circles.

EXAMPLE 6



p.813

FIGURE 1

* Boolean functions

(e.g. $F(x, y, z) = xy + \overline{z}$)

In general, let $B = \{0, 1\}$ $B^n = \{(x_1, x_2, \cdots, x_n) \mid x_i \in B\}$

How many possible Boolean function exists?

e.g. 1 bit input, 1 bit output (output is ALWAYS 1 bit) e.g. $F : B^3 \to B$

x	F_1	F_2	F_3	F_4
1	0	0	1	1
0	0	1	0	1
	$=0$	$=\overline{x}$	$=x$	$=1$

in general, n bits of input $\Rightarrow 2^n$ rows

$\Rightarrow$ columns each also have 2^n entries (each entry is a bit)

$\Rightarrow$ requires $\underbrace{2^{(2^n)}}_{\#\text{ boolean } f^{cn}\text{s}}$ columns

p.814 the case of 2 bit input $\# F^{cn} = 2^{2^2} = 2^4 = 16$

TABLE 3 The 16 Boolean Functions of Degree Two.

x	y	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Most "normal" algebra rules apply directly

(e.g. Predince ( ), $\cdot$, $+$ complement is applied immediately after evaluation of underlying expression

e.g. $\underbrace{\underset{\text{at least 1T}}{(x+y)}\ \underset{\text{at least 1T}}{(a+b)}}_{\text{Both } T \to T} = xa + xb + ya + yb$

each of these terms "checks" one of 4 possible case of "there's at least one T inside each parenthesis"

e.g. $(x+y)(x+z) = xx + xz + xy + yz$

$= x(x+y+z) + yz$

$= x + yz$

Thm $x(x+y+z) = x$

prof 1)

expand $xx + xy + xz = x + x(y+z) = x\underbrace{\overset{\text{T or a is T}}{(1+y+z)}}_{T} = x$

prof 2) truth table

x	y	z	$x+y+z$	$x(x+y+z)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

* Representing Boolean Functions

Def: a literal is any Boolean variable (e.g. x), or its complement ($\overline{x}$)

Def: Given n literals $x_1, x_2, \cdots, x_n$, a minterm is a product containing every literal or its complement exactly once

e.g. $y_1 \cdot y_2 \cdot y_3 \cdots y_n$ where y_i is either x_i or $\overline{x_i}$

Def: a Boolean sum of minterms representing a f^{cn} is called the "sum of products" or "Disjunctive Normal Form (DNF)" p.821

Thm given n Boolean variable, every Boolean on them can be expressed as sum-of-products (i.e. DNF)

e.g. Find DNF for $F(x, y, z) = (x+y)\overline{z}$

$= x\overline{z} + y\overline{z}$

$= x(y+\overline{y})\overline{z} + (x+\overline{x})y\overline{z}$

$= xy\overline{z} + x\overline{y}\overline{z} + xy\overline{z} + \overline{x}y\overline{z}$

$= xy\overline{z} + x\overline{y}\overline{z} + x\overline{y}\overline{z} + \overline{x}y\overline{z}$

OR truth table

TABLE 2

x	y	z	$x+y$	$\overline{z}$	$(x+y)\overline{z}$
1	1	1	1	0	0
1	1	0	1	1	1
1	0	1	1	0	0
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	1	1
0	0	1	0	0	0
0	0	0	0	1	0

*Functional Completeness

defines some set of operators then can express any Boolean function

Thm the set of Boolean operators $\{+, \cdot, \bar{\ }\}$ are functionally complete

Note: We can eliminate "+" by De Morgan's Law

$x + y = \overline{\bar{x} \cdot \bar{y}} \Rightarrow \{\cdot, \bar{\ }\}$ is F.C.

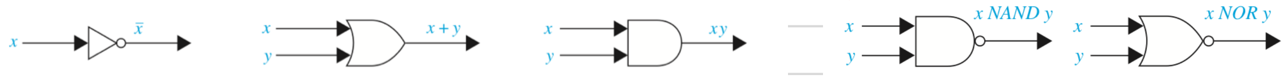
Can we find smaller set of operators that is F.C?

e.g. $NAND(x \cdot y) = \overline{x \cdot y}$, turns out NAND itself, is F.C.

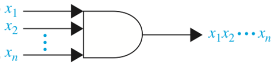
*Logic gates (Circuit Diagrams) p.823

Feb 28, 2017

p.828



(a) Inverter (b) OR gate (c) AND gate

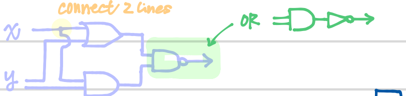


(AND)

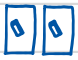
FIGURE 2 Gates with *n* Inputs.

combinational circuits gating network

e.g. $\overline{(x+y) \cdot (xy)}$



e.g. Design goal = create 2-switch light ∴ on with

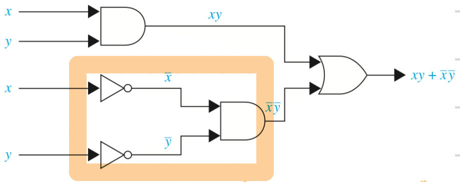
or

p.825

$F(x,y) = xy + \bar{x}\bar{y}$

$\Rightarrow$ it's off when $\bar{x}y + x\bar{y}$

TABLE 1

x	*y*	*F*(*x*, *y*)
1	1	1
1	0	0
0	1	0
0	0	1



this is NOT "NAND"

* Binary Addition

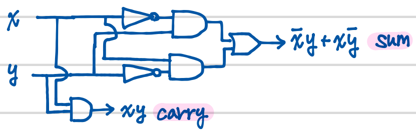
e.g. $11011 + 01001$

	1			1	1	← carry
	1	1	0	1	1	using Half Adder to get {sum & carry
+	0	1	0	0	1	
1	0	0	1	0	0	← sum (for each digit)

p.826

TABLE 3 Input and Output for the Half Adder.

Input		*Output*	
x	*y*	*s*	*c*
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	0



Half Adder

p.827

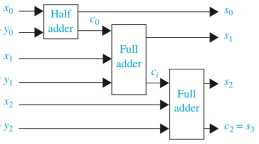


FIGURE 10 Adding Two Three-Bit Integers with Full and Half Adders.

TABLE 4 Input and Output for the Full Adder.

Input			*Output*	
x	*y*	c_i	*s*	c_{i+1}
1	1	1	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1
0	1	0	1	0
0	0	1	1	0
0	0	0	0	0

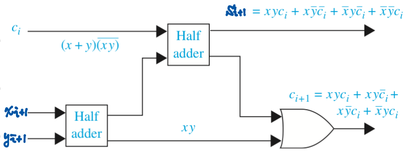


FIGURE 9 A Full Adder.

*Binary Subtractor

Half: Difference $= x \oplus y$

Borrow $= \bar{x}y$

Full: Difference $= x_{i+1} \oplus y_{i+1} \oplus B_i$

Borrow $= B_{i+1} = \overline{x_{i+1}}B_i + \overline{x_{i+1}}y_{i+1} + y_{i+1}B_i$

* Languages and Grammars p847

int i; symbol token

i = 2; statement / sentence phrase

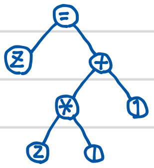
$i = 2 * i + 1;$ expression

Syntax = form of an expression (we don't care if the statement is nonsensical)

Semantix = assigns meaning ↳ But we care if a statement is valid

* parse tree (derivation tree)

e.g, $z = 2 * 1 + 1$



e.g, p,854

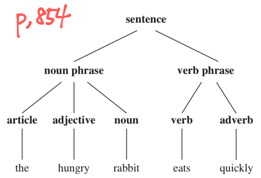


FIGURE 1 A Derivation Tree.

Def = Grammar describes a language by describing syntactically valid sentences (phrase/statement)

Def = a set of valid statements describes a language

Def = tokens (aka terminals) are atomic (i.e, smallest meaningful strings)

Def = symbols describes parts of sentences and can be terminal or non-terminal

e.g, (English sentence)

sentence ⟶ noun phrase, verb phrase

Here, this means "can be expressed" or "produces"

noun phrase ⟶ article, noun

article ⟶ "the" | "a"

OR

noun ⟶ "horse" | "rabbit"

Verb phrase ⟶ verb, adverb ← non-terminal

verb ⟶ "runs" | "eats"

adverb ⟶ "quickly" | "slowly"

terminals

symbols

Def = phrase-structure grammar $G = (V, T, S, P)$

March 2, 20

V = vocabulary ... a finite, nonempty set of elements called symbols

T = terminal (non-terminal $N = V \setminus T$)

P = productions (aka production rules)

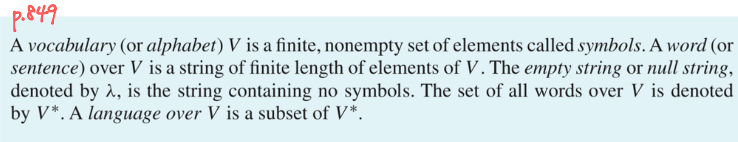
S = start symbol

e.g, $S \to \lambda \mid aSb$

OR

valid statements = $\lambda,\ ab,\ aabb,\ aaabbb,\ \dots$ $= \{a^n b^n \mid n = 0, 1, 2 \dots\}$ p.850

p.849



A *vocabulary* (or *alphabet*) V is a finite, nonempty set of elements called *symbols*. A *word* (or *sentence*) over V is a string of finite length of elements of V. The *empty string* or *null string*, denoted by λ, is the string containing no symbols. The set of all words over V is denoted by V^*. A *language over* V is a subset of V^*.

Def = λ, the empty string, is the string containing no symbols

Let $W_0 = l\, z_0\, r$ be a string of symbols (which could be terminal or non-terminal)

$W_1 = l\, z_1\, r$ be a string of symbols (which could be terminal or non-terminal)

Def = if ∃ production s.t. $z_0 \to z_1$, then we say string W_1 is directly derivable from W_0, written $W_0 \Rightarrow W_1$

e.g, $\underbrace{\text{article}}_{z_0}\ \underbrace{\text{noun}}_{r}$ W_0 $\Rightarrow$ $\underbrace{\text{"the" | "a"}}_{z_1}\ \underbrace{\text{noun}}_{r}$ W_1

$l = \lambda$ $l = \lambda$

Def = $W_0 \Rightarrow W_1 \Rightarrow W_2 \Rightarrow \cdots \Rightarrow W_n$, we say W_n is indirectly derivable from W_0, written $W_0 \overset{*}{\Rightarrow} W_n$,

and the sequence of steps, $W_0 \Rightarrow W_1 \Rightarrow W_2 \Rightarrow \cdots \Rightarrow W_n$, is called a derivation

Def= given G, $\mathcal{L}(G)$ is the set of all valid sentences derivable from G, aka the Language defined by G

Note= There can be more than one derivation showing $W_0 \overset{*}{\Rightarrow} W_n$

Note= Most language can be described by many grammars : $\forall \mathcal{L}\ \exists^{+} G\ \ \mathcal{L}(G)=\mathcal{L}$ Professor's Def: definitely more than 1

*Types of grammars (restrict what types of productions are allowed)

Type 1 Context-sensitive grammars : $lAr \to lwr$ (you can only replace A with w as long as between l&r

"A can only produce w if surrounded on both sides by l and r, i.e, l and r provide the only context in which A can be replaced with w"

Type 2 Context-free grammars : $A \to w$

" A can be replaced with w anytime"

Type 3 "Regular" grammars (only 3 types of productions are allowed)

i) $A \to \lambda$

ii) $A \to a$

iii) $A \to aB$ w/ a=terminal & A, B are non-terminals

allows left-to-right parsing of input strings, reading exactly one terminal at a time

e.g, from previous page

sentence → "a" nvp | "the" nvp

nvp → "horse" vp | "rabbit" vp

vp → "runs" adv | "eats" adv

adv → "slowly" (E) | "quickly" (E)

(E → λ)

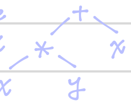
* simple example of mathematical expressions

e.g, $x+y$, $(x+y)*y$, $((x+y)*y)+x$

$E \to (E) \mid E*E \mid E+E \mid V$

$V \to x \mid y$

parse tree "$(x*y)+x$"



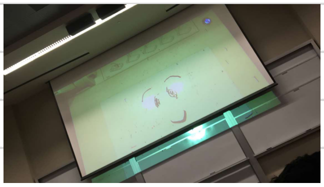
* Backus-Naur Form (BNF) p.853

ASCII method describing productions

<E> ::= (<<E>>) where < > is non-terminal

<V> ::= x | y

*Midterm review



March 7, 2017

Part A 4a) $A \cap B = \{4,5\}$ $\therefore P(A\cap B) = \{\emptyset, \{4\}, \{5\}, \{4,5\}\}$ power set always contains ∅ NULL

Part B $\exists n>0\ \ n=\sum_{i=1}^{n-1} i$ ←not "n" $n=3=2+1$ Since this is an existence proof, showing one example is enough

\#7 Hint = $(AB)^t = B^t A^t$

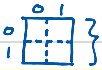
Def (of symmetric matrix): $AA^t = (AA^t)^t$

Let $B=A^t$ then $AB=(AB)^t = B^t A^t = AA^t \Rightarrow AA^t = (AA^t)^t$ ∎

The last Q (you don't need to prove it)

in this class, $\mathbb{N} = \mathbb{Z} \setminus \mathbb{Z}^-$

$\forall (n,x,y,z) \in \mathbb{N}^4\ [n>2 \to x^n + y^n \neq z^n]$

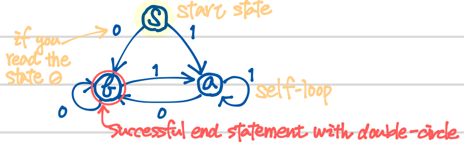
\#8  } 4 pairs $\therefore 2^4 = 16$ # of ordered pair

* Finite state machine

integer to bit string (base 2)

e.g. $6 = 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 \rightarrow 00110$

Goal = find FSM that recognizes base-2 integers that are even ⇒ (bit string) end with a zero



March 9, 2017

FSM = FSA

ending state can be more than 1 success state

transition (state) table

state	input	next state	output
S_0	0	b	
	1	a	
a	0	b	
	1	a	
b	0	b	
	1	a	

FSA / FSM … used for language recognition

Let V = vocabulary (tokens, terminals / non-terminals)

V^* = strings (not necessarily valid of vocab)

zero or more concatenated members of vocab

e.g. $\boxed{\text{int}}$ $\boxed{i}$ $\boxed{=}$ $\boxed{2}$ $\boxed{;}$ … 5 tokens valid string in C / Java

Let A, B are sets of strings from V^*, then $AB = \{xy \mid x \in A,\ y \in B\}$ p.866

$A, B \subseteq V^*$ e.g. A = {"input", "output"}, B = {"x", "y"}

string AB are "input x" "input y" "output x" "output y"

Def = Kleen closure of a set of strings A is

$A^* = \bigcup_{k=0}^{\infty} A^k$ A concatinated k times (Not multiplication)

Def = FSA / FSM = $M = (S, I, f, S_0, F)$

where S = set of states

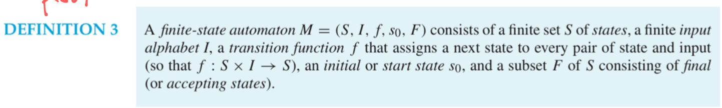
I = input alphabet

f = state transition table

S_0 = start state

F = set of successful final states

p.867

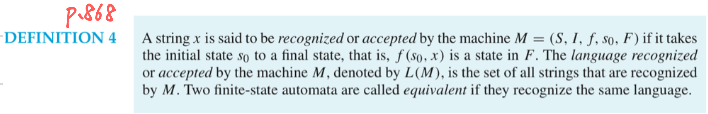

DEFINITION 3 A *finite-state automaton* $M = (S, I, f, s_0, F)$ consists of a finite set S of *states*, a finite *input alphabet* I, a *transition function* f that assigns a next state to every pair of state and input (so that $f : S \times I \rightarrow S$), an *initial* or *start state* s_0, and a subset F of S consisting of *final* (or *accepting states*).

Def = string $x = "x_0\ x_1\ x_2 \cdots x_n"$ $x_i \in$ terminals / tokens is recognized

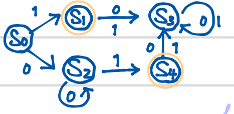
if machine M starting in state S_0 and reading entire x ends in a state in F,

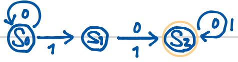
Note = Two FSMs are equivalent if they both recognize the same language

p.868


DEFINITION 4 A string x is said to be *recognized* or *accepted* by the machine $M = (S, I, f, s_0, F)$ if it takes the initial state s_0 to a final state, that is, $f(s_0, x)$ is a state in F. The *language recognized* or *accepted* by the machine M, denoted by $L(M)$, is the set of all strings that are recognized by M. Two finite-state automata are called *equivalent* if they recognize the same language.

e.g.





$\mathcal{L}(M_0) = \mathcal{L}(M_1)$

* NDFSMs (non-deterministic FSMs) … only mathematical construct

e.g. TSP (traveling salesman problem) = n cities to visit

goal = find the cheapest path that each city is once visited → n! possible pathes Note = $70! > 10^{100}$

in principle, what is the answer? (∃ a solution?)

→ the case of TSP, yes (∃ path for TSP cost < 100?)

nondeterministic polynomial time

Important Note = once TSP (NP-complete) is phrase as yes/ No, a "yes" answer is easily verified

Def: given a string $x = x_0 x_1 x_2 \cdots x_n$, NDFSM "M" recognizes x if ∃ a path through FSM ending a final state in F

Note: the state table for NDFSM has multiple "next" states

p-873

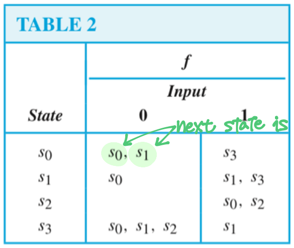


TABLE 2

State	f Input 0	f Input 1
s_0	s_0, s_1	s_3
s_1	s_0	s_1, s_3
s_2		s_0, s_2
s_3	s_0, s_1, s_2	s_1

next state is chosen

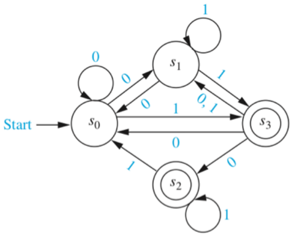


FIGURE 6 The Nondeterministic Finite-State Automaton with State Table Given in Table 2.

* Turing Machines

March 14, 2017

FSMs are finite, i.e. finite memory due finite tests

e.g. it cannot recognize $\{0^n 1^n \mid n \geq 0\}$ for arbitrary n (works fine for pre-known max n)

infinite in both directions

⇒ Turing Machines have "tape", which provides memory sourse

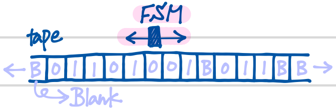
"idealized" computer, for mathematical description only

conceptialized

only operation allowed = read one cell, write one cell, move L or R by one cell

FSM is responsible for deciding what to write, and where to move

- What's order R/W head
- current state of FSM



"Turning Machines have read and write capabilities on the tape as the control unit moves back and forth along this tape, changing states depending on the tape symbol read (888)"

p.889

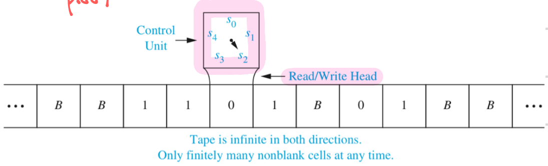


FIGURE 1 **A Representation of a Turing Machine.**

Def = Turing Machine $T = (S, I, f, S_0, F)$

where S = set of states in FSM

I = Input/Output symbols + "B" are allowed on the tape

$f = \underbrace{(S \times I)}_{\text{current state} \times i/o} \rightarrow \underbrace{(S \times I \times \{L, R\})}_{\text{new state} \times i/o \times d}$

S_0 = start state, F = final accepting states $\subseteq S$

at end "step" T,

① find a new state based on current state + input symbol

② write a new symbol on the tape

③ moves one cell left or right

We write this step as the five-tuple (s, x, s', x', d). If the partial function f is undefined for the pair (s, x), then the Turing machine T will *halt*.

A common way to define a Turing machine is to specify a set of five-tuples of the form (s, x, s', x', d). The set of states and input alphabet is implicitly defined when such a definition is used.

p.889

e.g. What is the final tape when the Turing machine T defined by the seven five-tuples $(s_0, 0, s_0, 0, R)$, $(s_0, 1, s_1, 1, R)$, (s_0, B, s_3, B, R), $(s_1, 0, s_0, 0, R)$, $(s_1, 1, s_2, 0, L)$ (s_1, B, s_3, B, R), and $(s_2, 1, s_3, 0, R)$ is run on the tape shown in Figure 2(a)?

since there is no five-tuple beginning with the pair of $(S_3, 0)$

Def= T recognizes a string x written on tape iff T, starts S_0, halts in a state in F

Note= if T halts on non-F state, or never halt, x is not recognized

Def= a language $\mathcal{L}$ is recognized by T iff x is recognized by T $\forall x \in \mathcal{L}$

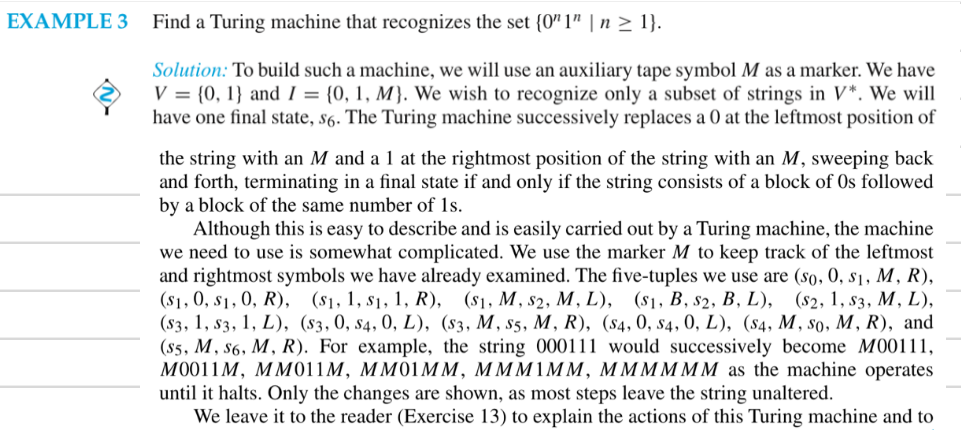
- given x as input, $T(x)$ replaces with y on tape
- We say $T(x)=y$ if T doesn't halt in non-F state, $T(x)$ is undefined
- many "extentions" e.g. multiple tapes, multiple R/W heads, multiple FSMs...
- almost never concerned about efficiency (except try not to take exponential time for something solvable in polynomial time)
  - if runtime on input of size is $\sim n^k$ for any fixed k, then polynomial time algorithm
  - if runtime on input of size is $\sim k^n$ for any fixed k, then exponential time algorithm

} $\forall k \exists n \; k^n > n^k$

* church-turing = anything that is computable, is computable by a Turing Machine ← thesis

any such machine is "Turing complete"

p.891

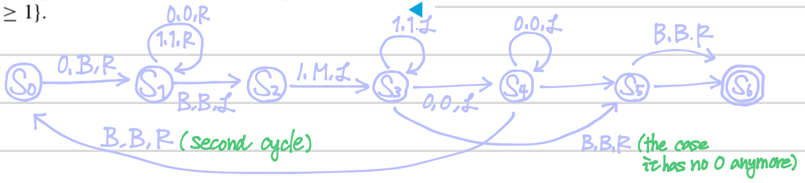


EXAMPLE 3 Find a Turing machine that recognizes the set $\{0^n 1^n \mid n \geq 1\}$.

Solution: To build such a machine, we will use an auxiliary tape symbol M as a marker. We have $V = \{0, 1\}$ and $I = \{0, 1, M\}$. We wish to recognize only a subset of strings in V^*. We will have one final state, s_6. The Turing machine successively replaces a 0 at the leftmost position of the string with an M and a 1 at the rightmost position of the string with an M, sweeping back and forth, terminating in a final state if and only if the string consists of a block of 0s followed by a block of the same number of 1s.

Although this is easy to describe and is easily carried out by a Turing machine, the machine we need to use is somewhat complicated. We use the marker M to keep track of the leftmost and rightmost symbols we have already examined. The five-tuples we use are $(s_0, 0, s_1, M, R)$, $(s_1, 0, s_1, 0, R)$, $(s_1, 1, s_1, 1, R)$, (s_1, M, s_2, M, L), (s_1, B, s_2, B, L), $(s_2, 1, s_3, M, L)$, $(s_3, 1, s_3, 1, L)$, $(s_3, 0, s_4, 0, L)$, (s_3, M, s_5, M, R), $(s_4, 0, s_4, 0, L)$, (s_4, M, s_0, M, R), and (s_5, M, s_6, M, R). For example, the string 000111 would successively become $M00111$, $M0011M$, $MM011M$, $MM01MM$, $MMM1MM$, $MMMMMM$ as the machine operates until it halts. Only the changes are shown, as most steps leave the string unaltered.

We leave it to the reader (Exercise 13) to explain the actions of this Turing machine and to explain why it recognizes the set $\{0^n 1^n \mid n \geq 1\}$.



* Complexity, Decidability, Computability

March 16, 2017

Def= a "decision problem" is a problem with a Y/N answer

Note= $\exists$ undecidable problems

e.g. halting problem = given & arbitrary problem p and input x, does p halt when applied to x?

Def= A problem is a decidable if $\exists$ a concreate algorithm that always decides it

check pdf= University of Waterloo
CS 360 Introduction to the
Theory of Computing
Winter 1998

Some easy-specified functions are not computable

e.g. given n, what is the longest possible finite string that can be output by Turing Machine w/ n states?

Note= specific small n, it's computable

e.g. "Hard problem" ... NP-complete

P = { set of all problems computable in polynomial time by Determistic T.M. }

NP = { set of all problems computable in polynomial time by non-Determistic T.M. }

Does P=NP? We can't disprove ⇒ We ASSUME P≠NP (not proved yet)