INTRO, LOGIC & PROOFS (slides)

ex: GCD of a & b

- basic algorithm ➝ check all numbers from a to 1 to get c
- runtime ➝ count # of divisions
    - worst case a & b coprime
    - c goes from a to 1 ➝ 2a divisions (not good large scale)
- better algorithm ➝ use euclid (with recursion)
    - if a=0 return b; else return gcd(b%a, a)

ex: stable matching

- “propose accept” algorithm
    - each m proposes to highest ranked w
    - if w likes the proposal more than current m, she breaks up and gets engaged to proposer
    - each m crosses off w who rejected him

ex: sorting

- roughly n(log(n)) comparisons

proposition: statement that is either true or false

- propositions can be combined using **logical operators** (and = ^)
- **truth table**: possibilities for **compound propositions**
    -
P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F
- basic connectors
    - and ∧, or ∨, not ⌐ $\overline{P}$
    - xor ⊻ $\oplus$ ➝ exactly 1, not both

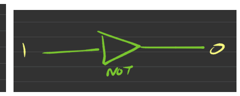
- $P \veebar Q = (P \wedge \neg Q) \vee (\neg P \wedge Q)$
- any compound proposition can be expressed using and, or, not
- to "evaluate" a compound proposition, plug in values T/F for each variables
- ex: truth table for $(\neg P \wedge Q) \vee (P \wedge Q) \vee (P \wedge \neg Q)$
    - is also equivalent to $\vee$

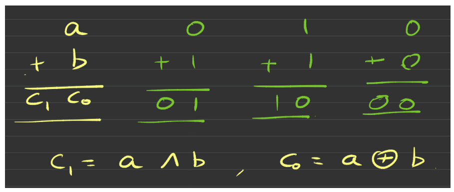
P	Q	expr
T T F F	T F T F	T T T F

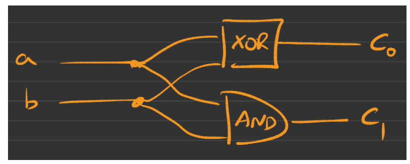
- **implication**: one being true means the other is true
    - P "implies" Q, $P \rightarrow Q$
    - implication is false ONLY when P is true and Q is false, essentially like $\neg P \vee Q$
    - $P \leftrightarrow Q$ is true if P and Q are both T or both F (logical equivalence = if and only if)
        - $P \leftrightarrow Q = (P \rightarrow Q) \wedge (Q \rightarrow P)$
- **De Morgan's laws**
    - $\neg(P \wedge Q) = \neg P \vee \neg Q$ → not(P or Q) = neither, not(P) and not(Q)
    - $\neg(P \vee Q) = \neg P \wedge \neg Q$ → not(P and Q) = either, not(P) or not(Q)
- **distributive law**
    - $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$
    - $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$
- can use laws to simplify formulas (like distributing P or Q and finding things always true/false)
- ex: simplify $\neg(A \wedge (B \vee C))$
    - $= \neg A \vee (\neg B \wedge \neg C) = (\neg A \vee \neg B) \wedge (\neg A \vee \neg C)$
- **valid**: always true $(P \vee \neg P)$
- **satisfiable**: not always false ($P \wedge \neg Q$, not $P \wedge \neg Q$) → go through truth table and wait for

logic applications

- circuits in hardware → 1=true, 0=false for bits of info
    - transistors → 



    - adding 2 one bit numbers
        - 



- logic in programs is boolean

PREDICATES (slides)

predicate: proposition quantified by a variable

- ex: P(n): n is even ➝ P(2) is true

variables

- domain: set from which variables are drawn

quantifiers

- ∀ = for all, ∃ = there exists
  - ex: ∀x ➝ $x^2 \geq 0$ (true)
- negation
  - ⌐(P(x)) is T if P(x) is F
  - ⌐(∀ x P(x)) ≡ ∃x ⌐(P(x))
    - (for all x, P(x) = T) is false ➝ there exists x for which P(x) = F
  - ⌐(∃x P(x)) ≡ ∀x ⌐(P(x))
    - (for some x, P(x) = T) is false ➝ for all x, P(x) = F
  - can also prove this with De Morgan's law
    - ⌐(∀ x P(x)) = $= \overline{P(x_1) \wedge P(x_2) \wedge \dots \wedge P(x_m)}$ $= \overline{P(x_1)} \vee \overline{P(x_2)} \vee \dots \vee \overline{P(x_m)}$
    - ⌐(∃x P(x) = $= \overline{P(x_1) \vee P(x_2) \vee \dots \vee P(x_m)}$ $= \overline{P(x_1)} \wedge \overline{P(x_2)} \wedge \dots \wedge \overline{P(x_m)}$
  - ex: Q(x,y) = T if x is a contestant m of show y; write no student has been on a tv show
    - ⌐(∃x ∃y Q(x,y))
    - ∀x ∀y ⌐(Q(x,y))

inference

- start with a set of valid propositions/predicates, write conclusions (other statements that are also valid/always true)
- rules of inference

    - modus ponens: $P, P \rightarrow Q, \therefore Q$
        - if P is true, and P implies Q, Q is true
        - $P \wedge (P \rightarrow Q) \Rightarrow Q$
    - modus tollens: $\neg Q, P \rightarrow Q, \therefore \neg P$
        - if Q is false, and P implies Q, P is false
        - if P = T, then since $P \rightarrow Q$, Q = T but Q = F, $\therefore$ P must be F
    - $P \rightarrow Q, Q \rightarrow R, \therefore P \rightarrow R$
    - $P \wedge Q, \therefore P$
    - $P \vee Q, \neg P, \therefore Q$
    - $P, Q, P \wedge Q$
    - to prove, use a truth table for P, Q, and whatever other statements
- inference with predicates
    - (see lecture slides for examples)
- **model**: assignment of meaning to predicates
    - inference is valid if it's T regardless of model
    - inference is invalid if there is a model where preconditions are T but conclusion is F
- axiom: a set of propositions we believe hold true
- theorem: another true statement deduced from axioms with rules of inference

predicate logic (recitation)

- variables no longer get assigned truth values → assigned elements from the domain
    - truth values depend on domain and predicate
- variables are quantified over range over some set (domain)
- notation:
    - $x \in A$ - x is an element of A
    - $\forall$ - for all
    - $\exists$ - there exists
    - $\wedge$ - and
    - $\vee$ - or
    - $\mathbb{N}$ - the set of natural numbers
    - $\mathbb{Z}$ - the set of integers
    - $\mathbb{Q}$ - the set of rational numbers
    - $\mathbb{R}$ - the set of real numbers

- mixing the order of quantifiers could result in different statements (not guaranteed equivalent)
    - ok when there are multiple quantifiers of the same kind next to each other (like existential vs universal)

PROOFS (slides, slides)

recap

- **axiom**: a set of propositions we believe hold true
- **theorem**: another true statement deduced from axioms with rules of inference

proof strategies

- direct proofs (typical proofs with a progression of true statements)
- cases (prove true for all cases)
- contrapositive (if trying to prove $P \rightarrow Q$, prove the contrapositive)
  - **contrapositive**: $\neg Q \rightarrow \neg P$
- contradiction (suppose the statement is false, go until contradicts)
- well ordering
  - **well-ordering principle (WOP)**: any set of positive integers has a smallest element
  - assume minimum counterexample
  - show smaller counterexample
- induction (show a predicate $P(n)$ is true for all natural numbers)
- to show $P \leftrightarrow Q$, show $P \rightarrow Q$, $Q \rightarrow P$ (to show $P \leftrightarrow Q \leftrightarrow R$, show $P \rightarrow Q \rightarrow R \rightarrow P$)
- common pitfalls
  - circular reasoning (ex: $P \rightarrow Q$ and $Q \rightarrow P$, $\therefore$ P is true)
  - proof by example (ex: $P(1)$ is true so $P(x)$ is true $\forall x \geq 1$)
  - proof by obviousness
  - obfuscation by notation (too much notation is confusing)

induction

- base case (show $P(1)$ is true)
- inductive case (show $P(n) \rightarrow P(n+1)$ for all $n \geq 1$)
- do not show $P(n+1) \rightarrow P(n)$

- strong induction: use truth of all P(1), P(2)...P(n) to show truth of P(n+1)
    - multiple base cases

Principle of Strong Induction.

Let P be a predicate on nonnegative integers. If

- $P(0)$ is true, and
- for all $n \in \mathbb{N}$, $P(0), P(1), \ldots, P(n)$ *together* imply $P(n+1)$,

then $P(m)$ is true for all $m \in \mathbb{N}$.

- structural induction: for recursive cases, use truth of P(n-q) to show P(n)

The Principle of Structural Induction.

Let P be a predicate on a recursively defined data type R. If

- $P(b)$ is true for each base case element $b \in R$, and
- for all two-argument constructors $\mathbf{c}$,

$$[P(r) \text{ AND } P(s)] \text{ IMPLIES } P(\mathbf{c}(r, s))$$

for all $r, s \in R$,
and likewise for all constructors taking other numbers of arguments,

then

$$P(r) \text{ is true for all } r \in R.$$