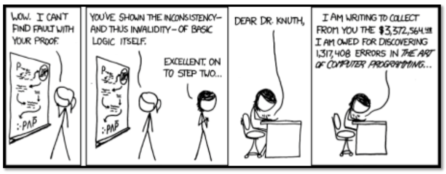
cse 311: foundations of computing

Spring 2015
Lecture 6: Predicate Logic, Logical Inference



turtles all the way down

If the tortoise walks at a rate of one node per step, and the hare walks at a rate of two nodes per step, then the distance between them increases by one node per step.

If the tortoise is on node x, and the hare is on node 2x, then the distance between them increases by one node per step.

OnNode(x)

Domain:
Non-negative Integers

nested quantifiers

- Bound variable names don't matter

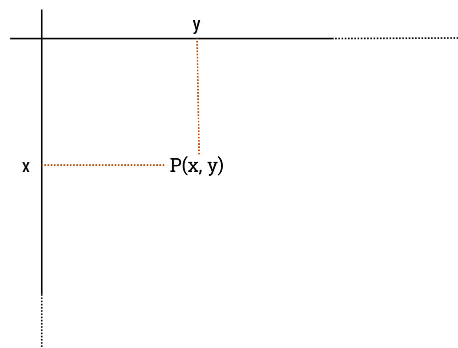
  $\forall x \exists y P(x, y) \equiv \forall a \exists b P(a, b)$

- Positions of quantifiers can sometimes change

  $\forall x (Q(x) \wedge \exists y P(x, y)) \equiv \forall x \exists y (Q(x) \wedge P(x, y))$

- But: order is important...

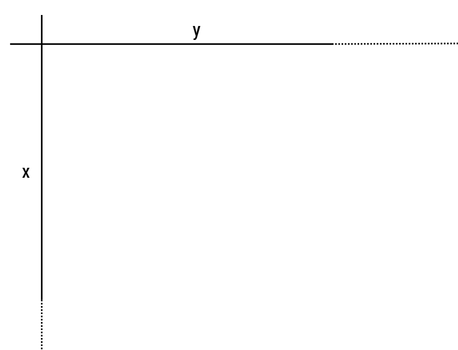
predicate with two variables



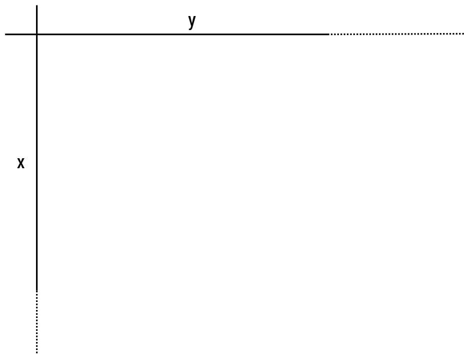
quantification with two variables

expression	when true	when false
$\forall x \forall y P(x, y)$		
$\exists x \exists y P(x, y)$		
$\forall x \exists y P(x, y)$		
$\exists x \forall y P(x, y)$		

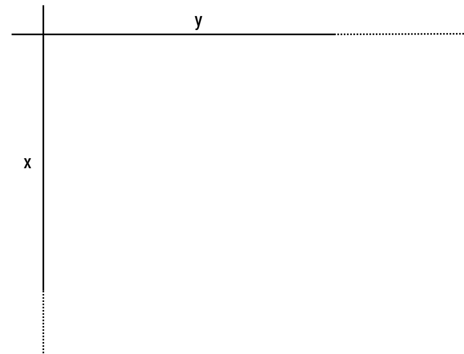
$\forall x \forall y\ P(x, y)$



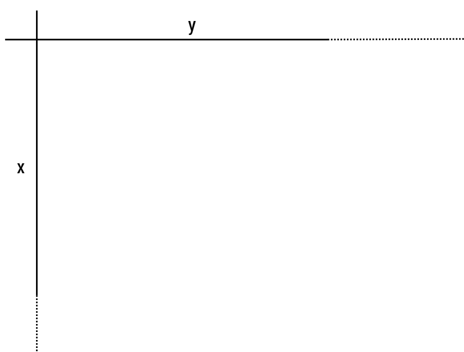
$\exists x\, \exists y\, P(x, y)$



$\forall x\, \exists y\, P(x, y)$



$\exists x\, \forall y\, P(x, y)$



quantification with two variables

expression	**when true**	**when false**
∀x ∀ y P(x, y)		
∃ x ∃ y P(x, y)		
∀ x ∃ y P(x, y)		
∃ x ∀ y P(x, y)		

logal inference

- So far we've considered:
  - How to understand and *express* things using propositional and predicate logic
  - How to *compute* using Boolean (propositional) logic
  - How to show that different ways of expressing or computing them are *equivalent* to each other
- Logic also has methods that let us *infer* implied properties from ones that we know
  - Equivalence is only a small part of this

applications of logical inference

- Software Engineering
  - Express desired properties of program as set of logical constraints
  - Use inference rules to show that program implies that those constraints are satisfied
- Artificial Intelligence
  - Automated reasoning
- Algorithm design and analysis
  - e.g., Correctness, Loop invariants.
- Logic Programming, e.g. Prolog
  - Express desired outcome as set of constraints
  - Automatically apply logic inference to derive solution

proofs

- Start with hypotheses and facts
- Use rules of inference to extend set of facts
- Result is proved when it is included in the set

an inference rule: *Modus Ponens*

- If p and $p \to q$ are both true then q must be true
- Write this rule as $\dfrac{p,\ p \to q}{\therefore\ q}$
- Given:
  - If it is Monday then you have a 311 class today.
  - It is Monday.
- Therefore, by modus ponens:
  - You have a 311 class today.

proofs

Show that r follows from p, $p \to q$, and $q \to r$

1.	p	given
2.	$p \to q$	given
3.	$q \to r$	given
4.	q	modus ponens from 1 and 2
5.	r	modus ponens from 3 and 4

proofs can use equivalences too

Show that $\neg p$ follows from $p \to q$ and $\neg q$

1.	$p \to q$	given
2.	$\neg q$	given
3.	$\neg q \to \neg p$	contrapositive of 1
4.	$\neg p$	modus ponens from 2 and 3

inference rules

- Each inference rule is written as: $\dfrac{A,\ B}{\therefore\ C, D}$

  ...which means that if both A and B are true then you can infer C and you can infer D.
  - For rule to be correct $(A \wedge B) \to C$ and $(A \wedge B) \to D$ must be a tautologies
- Sometimes rules don't need anything to start with. These rules are called axioms:
  - e.g. *Excluded Middle Axiom* $\dfrac{\quad}{\therefore\ p \vee \neg p}$

simple propositional enference rules

Excluded middle plus two inference rules per binary connective, one to eliminate it and one to introduce it:

$$\dfrac{p \wedge q}{\therefore\ p,\ q} \qquad \dfrac{p,\ q}{\therefore\ p \wedge q}$$

$$\dfrac{p \vee q,\ \neg p}{\therefore\ q} \qquad \dfrac{p}{\therefore\ p \vee q,\ q \vee p}$$

$$\dfrac{p,\ p \to q}{\therefore\ q} \qquad \dfrac{p \Rightarrow q}{\therefore\ p \to q}$$

Direct Proof Rule
Not like other rules

important: applications of inference rules

- You can use equivalences to make substitutions of any sub-formula.
- Inference rules only can be applied to whole formulas (not correct otherwise)

e.g. 1. $\cancel{p \rightarrow q}$ ~~given~~
2. $\cancel{(p \vee r) \rightarrow q}$ ~~intro $\vee$ from 1.~~

Does not follow! e.g . p=**F**, q=**F**, r=**T**

direct proof of an implication

- $p \Rightarrow q$ denotes a proof of q given p as an assumption
- The direct proof rule:
  If you have such a proof then you can conclude that $p \rightarrow q$ is true

Example:

proof subroutine

1.	p	**assumption**
2.	$p \vee q$	intro for $\vee$ from 1
3.	$p \rightarrow (p \vee q)$	direct proof rule