# Discrete Mathematics

*Alexander Pasko, Evgenii Maltsev, Dmitry Popov*

# Unit materials

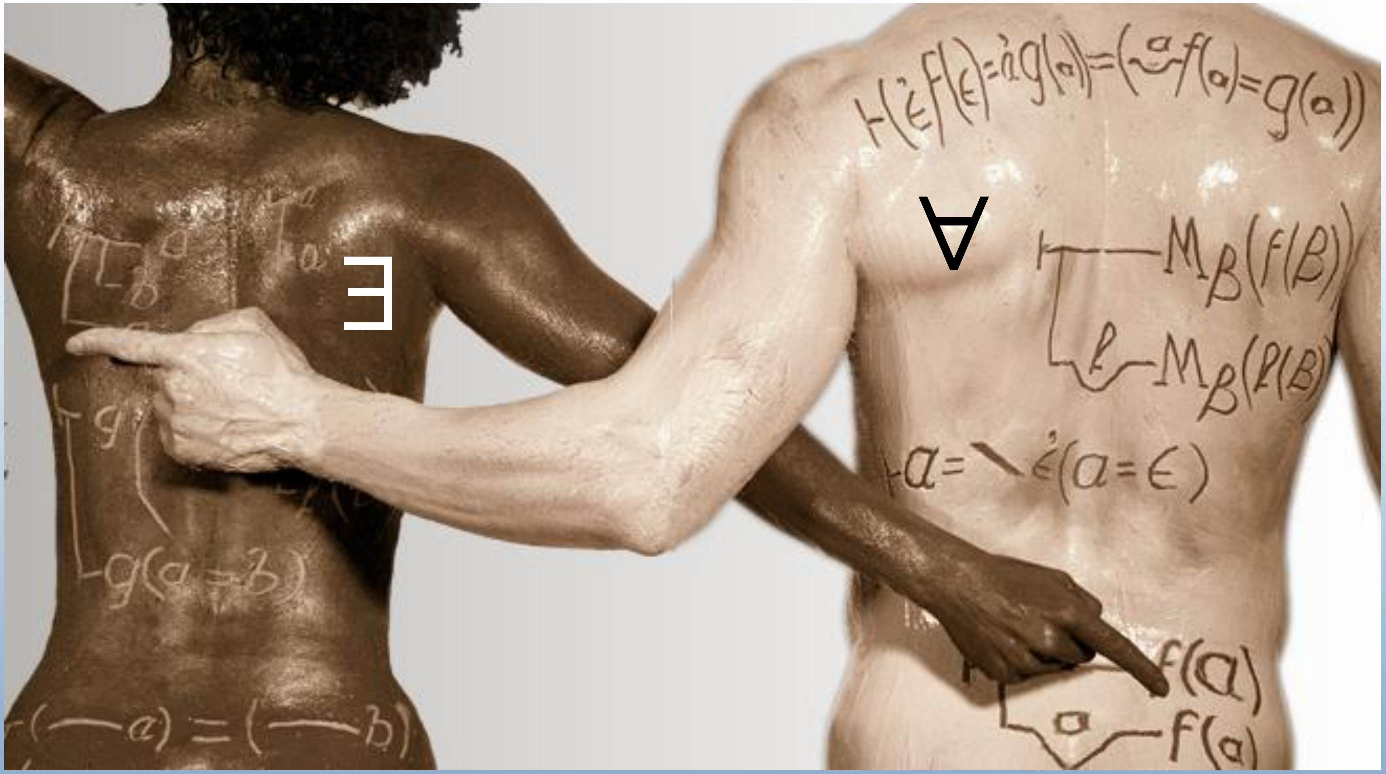
- Lecture notes
- Seminar handouts

are available at

http://gm.softalliance.net/

Advice: download and print lecture notes before the next lecture

# *Predicate Logic*




Cover fragment of "*Logic - Basics and Beyond*" *by G. Davies et al.,* Sine Metu

# Contents

- Predicate
- Quantifier expressions
- Nested quantifiers
- Applications of predicate logic
- Proofs

# Predicate Logic Background

- Propositional logic is not powerful enough to represent all types of statements or to express certain types of relationships between propositions.
- Example: "x is greater than 1“

  is not a proposition because you can not tell whether it is true or false unless you know the value of variable x.

# Predicate Logic Background

- Some logical equivalences can not be proven by the propositional logic:
  "Not all birds fly" is equivalent to
  "Some birds don't fly".
  "Not all integers are even" is equivalent to
  "Some integers are not even".
- For inferences like this, we need a more expressive logic

  Needed: treatment of \`some’ and \`every’

# Predicate Logic

- Predicate logic is an extension of propositional logic that permits concisely reasoning about whole classes of entities (presented by variables).
- Propositional logic treats simple propositions (sentences) as atomic entities.
- In contrast, predicate logic distinguishes the subject of a sentence from its property (presented by a predicate).

# Predicate Logic

- Sentence “The dog is sleeping” has two parts:
  – The phrase “the dog” denotes the subject - the object or entity that the sentence is about.
  – The phrase “is sleeping” denotes the predicate - a property that is true **of** the subject.
- Statement “z is greater than 7” has two parts:

  - z is the variable representing the subject

  - “is greater than 7“ is the predicate representing the property that the subject can have.

True or false? Not known: **not a proposition**

# Predicate Logic

Denote P(z)=“z is greater than 7“

P is a predicate “ is greater than 7“

z is the variable.

P(z) is the value of the propositional function P at z

Assign value to z, P(z) becomes a proposition

- Truth value of P(5) is …
- Truth value of P(8) is …

# Predicate

- Predicate is a verb phrase template that describes a property of objects, or a relationship among objects represented by the variables.

  Examples:

  "The car Tom is driving is blue"

  "The sky is blue"

  "The cover of this book is blue"

- A predicate is modeled as a *function* $P(\cdot)$ from objects to propositions.

  Phrase P**(x)="x is blue"** is a predicate and it describes the property of variable **x** being blue.

Examples:

"John gives the book to Mary",

"Jim gives a loaf of bread to Tom“

"Jane gives a lecture to Mary“

Predicate ***G( x, y, z )= “x gives y to z“***

Predicate logic generalizes the notion of a predicate to include propositional functions of **any** number of arguments.

Statement involving n variables can be denoted by

$$P\,(x_1, x_2, \bullet \bullet \bullet, x_n)$$

which is the value of the propositional function P at the n-tuple $(x_1, x_2, \bullet \bullet \bullet, x_n)$ and P is also called a n-place predicate or n-ary predicate.

# Predicate

- Convention.
  \- Lowercase variables *x*, *y*, *z*... denote classes of objects/entities;
  \- Individual constants denote individual objects: *a*, *b*, *c*,…
  \- Uppercase variables *P*, *Q*, *R*… denote propositional functions (predicates).

Result of applying a predicate P to a constant a is the proposition P(a)

Meaning: the object denoted by *a* has the property denoted by *P*.

# Universe of Discourse

- Power of distinguishing objects from predicates is that it lets you state things about many objects at once.
- Example: Let $P(x)$="$x+1>x$".

  Then "For any number $x$, $P(x)$ is true" instead of $(\mathbf{0}+1>\mathbf{0}) \wedge (\mathbf{1}+1>\mathbf{1}) \wedge (\mathbf{2}+1>\mathbf{2}) \wedge \ldots$
- What does "any" actually means?

  What are the limits on $x$ values?

# Universe of Discourse

- Proposition can be true for all values of a variable in a particular domain called
  - the domain of discourse
  - the universe of discourse (u.d.)
  - the domain.

Example:

$P(x)$=“$x+1>x$” is true for any $x$ in the domain of real numbers.

# Geometric interpretation

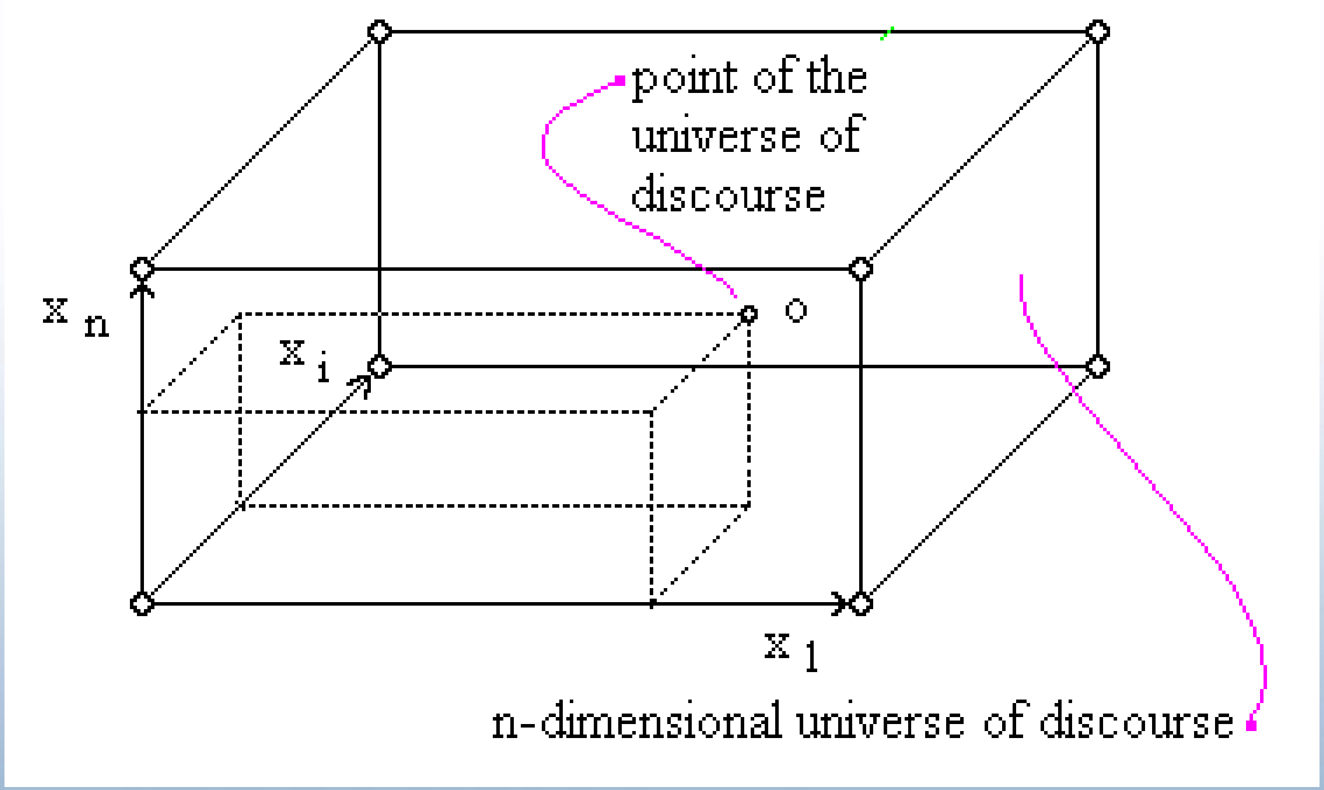



*Image by A. Savinov*

Scene bounding box represents the universe of discourse (can be entire space) and a point represents a variable value.

# Contents

- Predicate
- Quantifier expressions
- Nested quantifiers
- Applications of predicate logic
- Proofs

# Quantifier Expressions

- Quantifiers provide a notation that allows us to quantify (count) how many objects in the universe of discourse satisfy the given predicate.
- Quantification expresses the extent to which a predicate is true over a range of elements.
- In English, the words *all, some, many, none*, and *few* are used in quantifications.

# Universal Quantifier ∀

- "∀" is the FOR ∀LL or universal quantifier. $\forall x\, P(x)$ means *for all* x in the u.d., *P* holds true.
- Equivalence law for the ∀ quantifier:

if u.d. = a,b,c,…

$\forall x\, P(x) \Leftrightarrow P(\mathrm{a}) \wedge P(\mathrm{b}) \wedge P(\mathrm{c}) \wedge \ldots$

Example:

if x is real number, $\forall x\ (x+1>x)$ is true

# Universal Quantifier ∀ Example

Let u.d. of x be parking spaces at university.
Let $P(x)$ be predicate "$x$ is full"
Then universal quantification of $P(x)$, $\forall x\, P(x)$, is proposition:

- "All parking spaces at the university are full." or
- "Every parking space at the university is full." or
- "For each parking space at the university, that space is full."

# Counterexample

- $\forall x\, P(x)$ is false if and only if $P(x)$ is not always true when x is in the domain:

$$\neg(\forall x\, P(x)) \Leftrightarrow \neg(P(a) \wedge P(b) \wedge P(c) \wedge \ldots)$$
$$\Leftrightarrow \neg P(a) \vee \neg P(b) \vee \neg P(c) \vee \ldots$$

- One way to show that $P(x)$ is not always true is to find a *counterexample* – a value $x_0$ where $P(x_0)$ is false
- A single counterexample is enough to show that $\forall x\, P(x)$ is false

# Counterexample

## Counterexample 1:

P(x) = “x is black”

For u.d. of cats, $\not\forall x\ P(x)$ is false, see counterexample

## Counterexample 2:

Let $P(x) = x<2$

$\forall x\ (x<2)$ is false, because there is $x_0 = 3$,

where $P(x_0) = (3<2)$ is false.

# Existential Quantifier ∃

- "∃" is the ∃XISTS or existential quantifier. $\exists x\, P(x)$ means there *exists* an $x$ in the u.d. such that $P(x)$ is true.
- $\exists x\, P(x)$ is read as
  "There is an $x$ such that $P(x)$"
  "There is at least one $x$ such that $P(x)$"
  "For some $x$, $P(x)$."

# Existential Quantifier ∃

- Equivalence law for the ∃ quantifier:

if u.d. = a,b,c,…

$$\exists x\, P(x) \Leftrightarrow P(\mathrm{a}) \vee P(\mathrm{b}) \vee P(\mathrm{c}) \vee \ldots$$

Example:

if x is real number, $\exists x\ (x>3)$ is true,

because there is x=4 and 4>3 is true.

# Existential Quantifier $\exists$

- $\exists x\ P(x)$ is false if and only if $P(x)$ is not true for all x is in the domain:

$$\neg(\exists x\ P(x)) \Leftrightarrow \neg(P(a) \vee P(b) \vee P(c) \vee \ldots)$$
$$\Leftrightarrow \neg P(a) \wedge \neg P(b) \wedge \neg P(c) \wedge \ldots$$

- Loop search for true value:

  To see whether $\exists x\ P(x)$ is true, we loop through all the values of x searching for a value for which $P(x)$ is true. If we find one, then $\exists x\ P(x)$ is true. If we never find such an x, then we have determined that $\exists x\ P(x)$ is false.

# Existential Quantifier ∃ Example

Let u.d. of x be parking spaces at university.
Let $P(x)$ be predicate "$x$ is full."
Then the existential quantification *of* $P(x)$, $\exists x\, P(x)$, is the proposition:

- "Some parking space at the university is full."
- "There is a parking space at the university that is full."
- "At least one parking space at the university is full."

# Quantifier Equivalence Laws

- Definitions of quantifiers: If u.d.=a,b,c,…
  $\forall x\, P(x) \Leftrightarrow P(\mathrm{a}) \wedge P(\mathrm{b}) \wedge P(\mathrm{c}) \wedge \ldots$
  $\exists x\, P(x) \Leftrightarrow P(\mathrm{a}) \vee P(\mathrm{b}) \vee P(\mathrm{c}) \vee \ldots$
- From those, we can prove the laws:
  $\forall x\, P(x) \Leftrightarrow \neg\exists x\, \neg P(x)$
  $\exists x\, P(x) \Leftrightarrow \neg\forall x\, \neg P(x)$

❖ Which propositional equivalence laws can be used to prove this?

DeMorgan's

# Negation of Quantifiers

## De Morgan's Laws for Quantifiers

| *Negation* | *Equivalent Statement* | *When Is Negation True?* | *When False?* |
|---|---|---|---|
| $\neg\exists x P(x)$ | $\forall x \neg P(x)$ | For every $x$, $P(x)$ is false. | There is an $x$ for which $P(x)$ is true. |
| $\neg\forall x P(x)$ | $\exists x \neg P(x)$ | There is an $x$ for which $P(x)$ is false. | $P(x)$ is true for every $x$. |

Example:

"Every student in your class has taken a course in calculus." $\forall x\, P(x)$

$\neg\forall x\, P(x) \Leftrightarrow \exists x\, \neg P(x)$ means

"It is not the case that every student in your class has taken a course in calculus."

$\Leftrightarrow$

"There is a student in your class who has not taken a course in calculus."

# Free and Bound Variables

- Expression like $P(x)$ is said to have free variable x (meaning, $x$ is undefined).
- When quantifier (either ∀ or ∃) operates on expression which has one or more free variables, it binds one or more of those variables, to produce an expression having one or more bound variables.
- Variable also became bound when we assign value to this variable.

# Example of Binding

Example of binding variables using quantifiers:

- $P(x,y)$ has 2 free variables, $x$ and $y$.
- $\forall x\ P(x,y)$ has 1 free variable, and one bound variable. [Which is which?]
- “ S(z), where z=3 ” is another way to bind z.
- Expression with <u>zero</u> free variables is an actual proposition.
- Expression with <u>one or more</u> free variables is still only a predicate: let $Q(y) = \forall x\ P(x,y)$

# Contents

- Predicate
- Quantifier expressions
- Nested quantifiers
- Applications of predicate logic
- Proofs

# Nested Quantifiers

Two quantifiers are nested if one is within the scope of the other

Example: $\forall x\,(\exists y(x + y = 0))$ or simply
$\forall x\,\exists y(x + y = 0)$

can be considered $\forall x\ Q(x)$,

where $Q(x)$ is $\exists y P(x, y)$,

$P(x, y)$ is $x + y = 0$.

In English: for every real number x there is a real number y such that $x + y = 0$,

and $y = -x$.

# Nested Quantifiers

Different order of quantifiers:

$$\exists y\ \forall x\ (x + y = 0)$$

"There is a real number y such that for every real number x, x + y =0."

- Truth value is … ?

# Nested Quantifiers Example

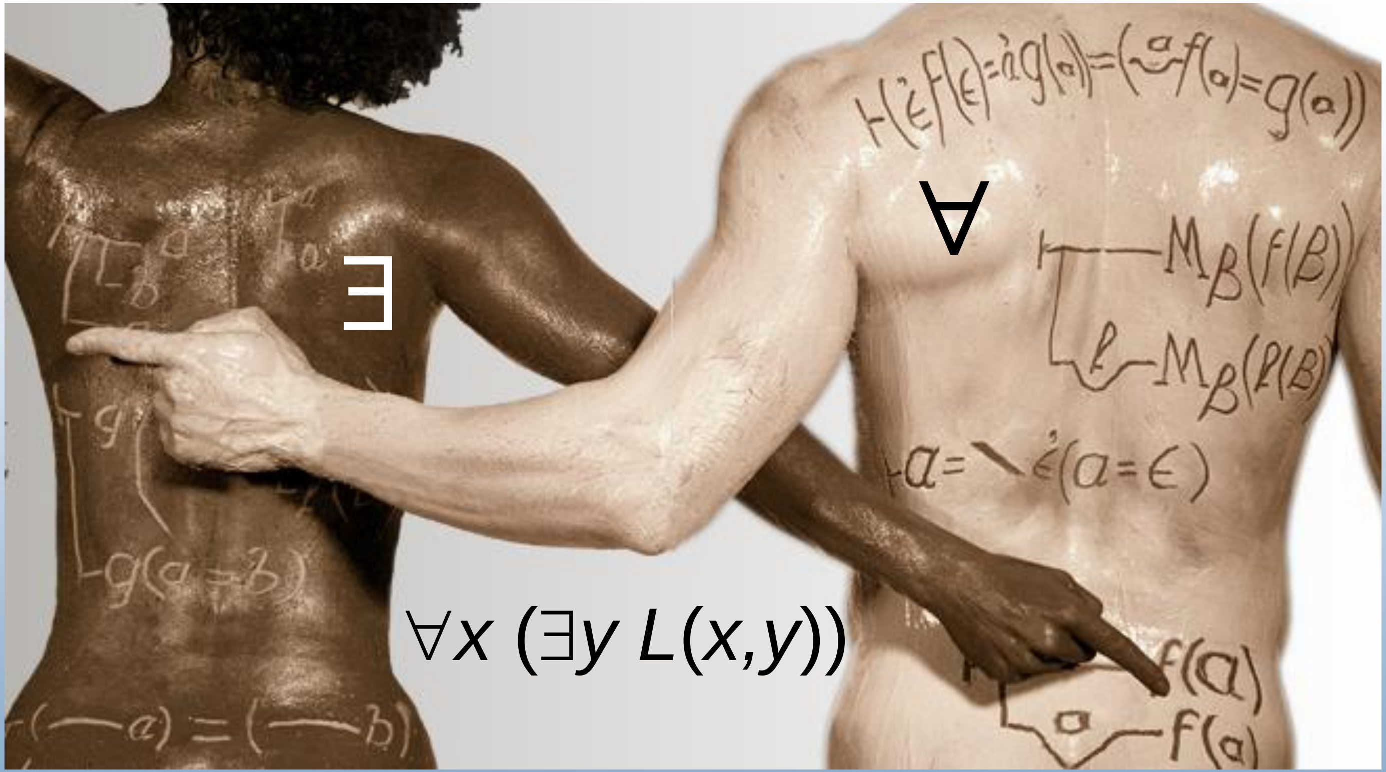
- Example: Let the u.d. of $x$ & $y$ be people.
- Let $L(x,y)$="$x$ likes $y$" (a predicate with 2 free variables)
- Then $\exists y\, L(x,y)$ = "There is someone whom $x$ likes." (a predicate with 1 free variable, $x$)

❖ Then $\forall x\, (\exists y\, L(x,y))$ = "Everyone has someone whom they like."

(A __Proposition__ with __0__ free variables.)

# Nested Quantifiers Example

“Everyone has someone whom they like.”




Cover fragment of “*Logic - Basics and Beyond” by G. Davies et al.*, Sine Metu

# Nested Quantifiers Exercise

- If $R(x,y)$="$x$ relies upon $y$," express the following in unambiguous English:
- $\forall x(\exists y\, R(x,y))$= Everyone has *someone* to rely on.
- $\exists y(\forall x\, R(x,y))$= There's a poor overburdened soul whom *everyone* relies upon (including himself)!
- $\exists x(\forall y\, R(x,y))$= There's some needy person who relies upon *everybody* (including himself).
- $\forall y(\exists x\, R(x,y))$= Everyone has *someone* who relies upon them.
- $\forall x(\forall y\, R(x,y))$= *Everyone* relies upon *everybody*, (including themselves)!

# Quantifications of two Variables

| ***Statement*** | ***When True?*** | ***When False?*** |
|---|---|---|
| $\forall x \forall y P(x, y)$<br>$\forall y \forall x P(x, y)$ | $P(x, y)$ 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. |
| $\exists x \exists y P(x, y)$<br>$\exists y \exists x P(x, y)$ | There is a pair $x, y$ for which $P(x, y)$ is true. | $P(x, y)$ is false for every pair $x, y$. |

# Negation of Nested Quantifiers

Example: find the negation of the statement $\forall x\, \exists y\, (xy = 1)$ so that no negation precedes a quantifier.

$\neg \forall x\, \exists y\, (xy = 1) \Leftrightarrow$

$\exists x\, \neg \exists y\, (xy = 1) \Leftrightarrow$

$\exists x\, \forall y\, \neg (xy = 1) \Leftrightarrow$

$\exists x\, \forall y\, (xy \neq 1)$

- What is the x value which makes the true counterexample?

# Review of Predicate Logic

- Predicates $P$, $Q$, $R$, … are functions mapping objects $x$ to propositions $P(x)$.
- Universe of discourse
- Quantifiers:

  $\forall x\ P(x)$ :≡ "For all $x$'s, $P(x)$."
  $\exists x\ P(x)$ :≡ "There is an $x$ such that $P(x)$."
  $\neg\forall x\ P(x) \Leftrightarrow \exists x\ \neg\ P(x)$
- $\forall x\ \exists y\ L(x,y)$ =
  "Everyone has someone whom they like."

# Contents

- Predicate
- Quantifier expressions
- Nested quantifiers
- Applications of predicate logic
- Proofs

# Applications of Predicate Logic

- Predicate Logic is the formal notation for writing perfectly clear, concise, and unambiguous mathematical definitions, axioms, and theorems for *any* branch of mathematics.
- Predicate logic with function symbols, the “=” operator, and a few equivalence rules is sufficient for defining any mathematical system, and for proving anything that can be proved within that system!

# Applications of Predicate Logic

- It is the basis for clearly expressed formal specifications for any complex software system.
- It is basis for automatic theorem proving systems and many other Artificial Intelligence systems (automatic program verification).
- Predicate-logic like statements are supported by some of the more sophisticated database query engines.

# Predicates in Programming

Statement of some programming language:

*if (x > 0) then x := x + 1*

- Predicate *P(x)* = *"x > 0"*
- When this statement is encountered in a program, the value of the variable x is inserted into *P(x)*
- If *P(x)* is true for this value of *x*, the assignment statement *x := x + 1* is executed
- If *P(x)* is false for this value of *x*, the assignment statement is not executed, so the value of *x* is not changed
- *P(x)* can be very complex with Boolean operators

# Logic Programming

- There are some programming languages that are based entirely on predicate logic!
- The most famous one is called **`Prolog`**.
- A Prolog program is a set of propositions (“facts”) and (“rules”) in predicate logic.
- The input to the program is a “query” proposition:
  - Want to know if it is true or false.
- The Prolog interpreter does some automated deduction to determine whether the query follows from the facts.

# Example in Prolog

- Facts described in Prolog:

```
instructor(chan,math273)
instructor(patel,ee222)
instructor(grossman,cs301)
enrolled(kevin,math273)
enrolled(juana,ee222)
enrolled(juana,cs301)
enrolled(kiko,math273)
enrolled(kiko,cs301)
```

- New predicate:

```
teaches(P,S) :- instructor(P,C), enrolled(S,C)
```

means professor P teaches student S, comma is $\wedge$

- Queries in Prolog:

```
?enrolled(kevin,math273)
```

produces the response:

**`yes`**

```
?teaches(X,juana)
```

produces the response:

**`patel`**

**`grossman`**

# Contents

- Predicate
- Quantifier expressions
- Nested quantifiers
- Applications of predicate logic
- Proofs

# Nature & Importance of Proofs

- In mathematics, a proof is a correct (well-reasoned, logically valid) and complete (clear, detailed) argument that rigorously & undeniably establishes the truth of a mathematical statement.
- Why must the argument be correct & complete?

  - *Correctness* prevents us from fooling ourselves.
  - *Completeness* allows anyone to verify the result.
- Methods of mathematical argument (*i.e.*, proof methods) can be formalized in terms of rules of logical inference.

# Applications of Proofs

- An exercise in clear communication of logical arguments in any area of study.
- The fundamental activity of mathematics is the discovery through proofs of interesting new theorems.
- Theorem-proving has applications in program verification, computer security, automated reasoning systems, *etc.*
- Proving a theorem allows us to rely on its correctness even in the most critical scenarios.

# Proof Terminology

- Theorem
  A statement that has been proven to be true.
- Axioms, postulates, hypotheses, premises
  Assumptions (often unproven) defining the structures about which we are reasoning.
- Rules of inference
  Patterns of logically valid deductions from hypotheses to conclusions
  (i.e., equivalence laws)
- Theory
  The set of all theorems that can be proven from a given set of axioms.

# Basic Proof Methods

For proving *that q* follows from *p*, we have:

- *Direct* proof: Assume p is true, and prove q.
- *Indirect* proof: Assume $\neg$q, and prove $\neg$p.
- *Trivial* proof: Prove q by itself.

# Direct Proof Example

- Def. An integer $n$ is called *odd* iff
  $n=2k+1$ for some integer $k$;
  $n$ is *even* iff
  $n=2k$ for some $k$.
- Thm. (For all numbers $n$) If $n$ is an odd integer, then $n^2$ is an odd integer.
- Proof. If $n$ is odd, then $n = 2k+1$ for some integer $k$. Thus, $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$. Therefore $n^2$ is of the form $2j + 1$ (with $j$ the integer $2k^2 + 2k$), thus $n^2$ is odd.

# Indirect Proof Example

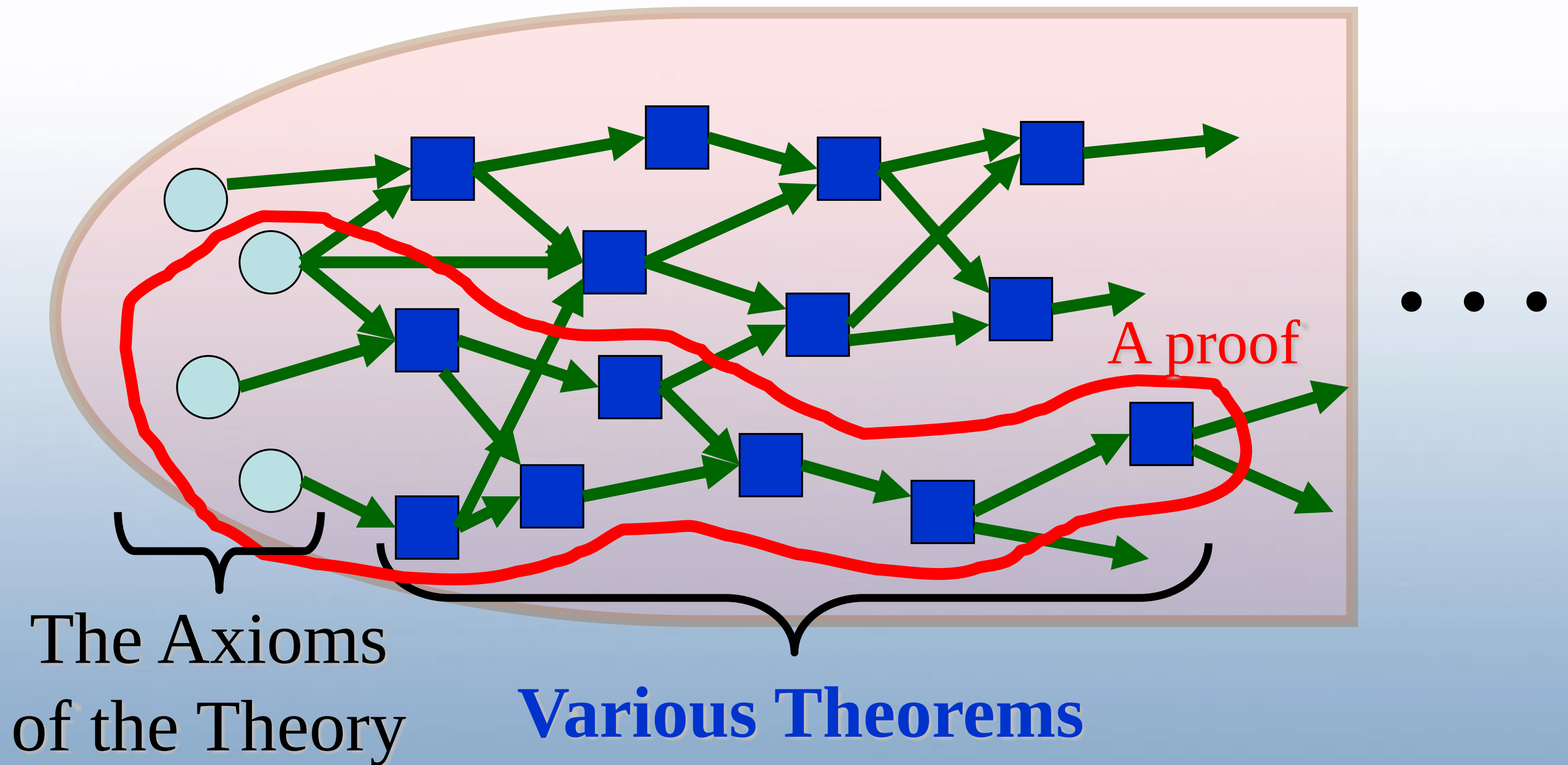
- Thm. (For all integers *n*)
  If $3n+2$ is odd, then *n* is odd.
- Proof. Suppose that the conclusion is false, *i.e.*, that *n* is even.
  Then $n=2k$ for some integer *k*.
  Then $3n+2 = 3(2k)+2 = 6k+2 = 2(3k+1)$.
  Thus $3n+2$ is even, because it equals $2j$ for integer $j = 3k+1$.
  So $3n+2$ is not odd.
  We have shown that $\neg(n \text{ is odd}) \rightarrow \neg(3n+2 \text{ is odd})$, thus its contra-positive $(3n+2 \text{ is odd}) \rightarrow (n \text{ is odd})$ is also true.

# Trivial Proof Example

- Thm. (For integers *n*) If *n* is the sum of two prime numbers, then either *n* is odd or *n* is even.
- Proof. *Any* integer *n* is either odd or even. So the conclusion of the implication is true regardless of the truth of the antecedent. Thus the implication is true trivially.

# Theory Structure

## A Particular Theory



*Questions?*