Course 50.050/50.550
Advanced Algorithms

Week 1 – Lecture L01.02

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Outline of Lecture

- Tuples, Cartesian products, set relations
- Propositional logic, operations on propositions
- Predicates, quantifiers, logical equivalence
- Axioms and axiomatic systems, consistency

Sequences and tuples

Recall: Elements in sets are unordered.

- Frequently, we want to consider elements in some order.
- **Question:** How do we introduce order? Use sequences!

A sequence is a well-defined **ordered** collection of objects.

- The objects in a sequence could repeat.
- The objects must be **indexable** by a subset of $\mathbb{Z}$.
- These objects are called terms or elements of the sequence.

A tuple is a **finite** sequence of objects.

- **Notation:** Given some finitely many objects, we have to place them in between round brackets ( ) to denote the tuple comprising these objects in the given indicated order.
- **Example:** The tuple of all students enrolled in this course, ordered according to the alphabetical order of their names:

- **Example:** The specific tuple $(95, \; 95, \; 91, \; 84, \; 68, \; 84)$.

Definition: Given $k \in \mathbb{N}$, a k-tuple is a tuple consisting of k objects.

Conventions for tuples

A tuple is **ordered**.

- **Example:** $(1,2,3)$ and $(3,2,1)$ are different tuples!

The objects contained in a tuple are called entries.

- **Example:** The tuple $(2,3,5)$ has three entries $2,3,5$.

The entries of a tuple can be sets or tuples or any object.

- **Example:** The tuple $(\{2,3\},(5,7))$ has two entries.
  - The first entry is a set, and the second entry is a 2-tuple.
- A 0-tuple (or an empty sequence) is an ordered collection with no objects, and so it is denoted by $\emptyset$ or $\varnothing$ or $()$.
  - Since there is no object, there is nothing to order!
- A 1-tuple (x) is sometimes called a singleton.
- A 2-tuple (x,y) is called an ordered pair.

Note: Just like sets, we can give names to tuples.

- **Example:** We can say "Let $v := (a,b,c)$ be a tuple." Then this tuple v is a 3-tuple. Its entries are a,b,c.
  - Common notation for tuples: Lowercase English/Greek letters.

Warning: Other (ambiguous) names for tuples: lists, vectors, arrays.

Ambiguity for lists, vectors, arrays: e.g. may require entries of the same type.

Sets of tuples from cartesian products of sets

Definition: Given any sets X and Y, the Cartesian product of X and Y, denoted by $X \times Y$, is the set

$$X \times Y := \{(x, y) | x \in X, y \in Y\}.$$

- The elements of the set $X \times Y$ are 2-tuples (i.e. ordered pairs).
- We allow the possibility that $X = Y$.

Definition: Given $k \in \mathbb{Z}^+$ and any k sets $X_1, \ldots, X_k$, the Cartesian product of $X_1, \ldots, X_k$, denoted by $X_1 \times \cdots \times X_k$, is the set

$$X_1 \times \cdots \times X_k := \{(x_1, \ldots, x_k) | x_1 \in X_1, \ldots, x_k \in X_k\}.$$

- The elements of the set $X_1 \times \cdots \times X_k$ are k-tuples.
- The k sets $X_1, \ldots, X_k$ are not necessarily distinct.
- When $X_1 = \cdots = X_k = X$, we can use the notation X^k.
- **Note:** If any X_i is $\emptyset$, then the Cartesian product equals $\emptyset$.
  - e.g. $\emptyset \times \mathbb{R} = \mathbb{R} \times \emptyset = \emptyset$.

Example: $\mathbb{R}^k$ is the Cartesian product of k copies of $\mathbb{R}$.

- $\mathbb{R}^k$ is also called the Euclidean k-space.
- We will study the geometry of Euclidean spaces in Week 12.

Binary relations

Suppose X and Y are sets.

Definition: A binary relation **from** X **to** Y is a subset of $X \times Y$.

- Also called a relation **from** X **to** Y, if the context is clear.

Intuition: Binary relations represent relationships between the elements of the two sets.

- If R is a binary relation from X to Y, and if $(x, y) \in R$, then we can interpret the ordered pair (x, y) as "x is related to y".

Example 1: A binary relation

$R \subseteq \Big\{$, , $\Big\} \times \{$'yellow', 'blue'$\}$

representing the relationship "**has the color of**" could be

$R = \Big\{\Big($, 'blue'$\Big), \Big($, 'yellow'$\Big), \Big($, 'yellow'$\Big), \Big($, 'blue'$\Big)\Big\}.$

Example 2: The binary relation $R \subseteq \{1, 2, 5\} \times \{2, 4, 6\}$ representing the relationship "**less than or equal to**" is

$$R = \{(1, 2), (1, 4), (1, 6), (2, 2), (2, 4), (2, 6), (5, 6)\}.$$

Functions as binary relations

Recall: Given any function $f : X \to Y$, both X and Y are sets, and every element $x \in X$ is mapped to some element $y \in Y$.

- Intuitively, every $x \in X$ is related to the element $f(x)$ in Y.
- **Consequence:** We can interpret a function $f : X \to Y$ as a binary relation R from X to Y, given by
$$R = \{(x, y) \in X \times Y : y = f(x)\}.$$
- With this interpretation, a function can be treated as a "particular kind" of binary relation.
  - Every $x \in X$ appears as a first entry of some ordered pair in R, and no two distinct ordered pairs in R have the same first entry.

Example 3: The square function $f : \mathbb{N} \to \mathbb{N}$ given by $x \mapsto x^2$ can be interpreted as a binary relation

$$R = \{(0,0), (1,1), (2,4), (3,9), (4,16), (5,25), \dots\} \subseteq \mathbb{N} \times \mathbb{N}$$

representing the relationship "**has the squared value equal to**".

Important Consequences:

- We can think of functions as sets. *(Be careful: functions are* ***not*** *sets!)*
- We can think of binary relations as generalizations of functions.

Notation for binary relations

Definition: Let X, Y be sets, and let R be a relation from X to Y.

- If $(x, y) \in R$, then we say that x is related to y by R, and we can use the notation $x\,R\,y$ to indicate this.
- If $(x, y) \notin R$, then we say that x is not related to y by R, and we can use the notation $x \not{R}\, y$ to indicate this.

Recall Example 2: The binary relation $R \subseteq \{1, 2, 5\} \times \{2, 4, 6\}$ representing the relationship "**less than or equal to**" is

$$R = \{(1, 2), (1, 4), (1, 6), (2, 2), (2, 4), (2, 6), (5, 6)\}.$$

- We can write: $1\ R\ 2, 1\ R\ 4, 1\ R\ 6, 2\ R\ 2, 2\ R\ 4, 2\ R\ 6, 5\ R\ 6$.
- **Remember:** R is a "name" of our binary relation.
  - It is a **symbol** representing our subset in $\mathbb{N} \times \mathbb{N}$.
- If we instead use the symbol "$\leq$" to denote our binary relation, then we have $1 \leq 2, 1 \leq 4, 1 \leq 6, 2 \leq 2, 2 \leq 4, 2 \leq 6, 5 \leq 6$.

This notation convention for writing the symbol R (or $\leq$) between the two objects that it "operates" on, is called the infix notation.

- **Examples of infix notation:** $x = y$, $A \subseteq B$, $\ell_1 \perp \ell_2$.

k-ary relations

Note: A binary relation can be treated as a relation on two sets.

- In general, we would want to consider relations on multiple sets.

Definition: Let $k \in \mathbb{N}$, and let $(X_1, \ldots, X_k)$ be a k-tuple of sets. A k-ary relation on $(X_1, \ldots, X_k)$ is a subset of $X_1 \times \cdots \times X_k$.

- We can give names to k-ary relations.
- Given a k-ary relation R on $(X_1, \ldots, X_k)$, the Cartesian product $X_1 \times \cdots \times X_k$ is called the domain of discourse for R.
- **Special names:** A 1-ary relation is a unary relation, a 2-ary relation is a binary relation, a 3-ary relation is a ternary relation.

Example: A 1-ary relation R representing "**is a perfect square**" is $R = \{0, 1, 4, 9, 16, 25, \ldots\}$, where the domain of discourse for R is $\mathbb{N}$.

Example: A 3-ary relation R representing "**consecutive integers**" is $R = \{\ldots, (-8, -7, -6), \ldots, (3, 4, 5), \ldots\}$, where the domain of discourse for R is $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$.

Remark on notation: In contrast to the notation for binary relations, we cannot use the in-fix notation to denote relations on k sets. Instead we use the usual set notation, e.g. if R is a 3-ary relation, then we could write $(x_1, x_2, x_3) \in R$.

What is logic?

Informally, logic is the "grammar" of mathematical reasoning.

- We follow rules of logic to make sense of algorithms and mathematical statements.
  - Is our code syntactically correct? How to tell if two algorithms always return the same outputs when given the same inputs?
  - When is a statement correct/incorrect? When is a statement true/false? Does the statement even make sense?
- We use logic to make inferences on relationships between different logical statements.
  - How do we systematically reason and make correct (logically sound) inferences? Why do we know that A implies B?
  - What possible steps of reasoning do we allow?

Logic is the study of the underlying logic of different "kinds" of logic.

- Analogous to how natural languages have different grammars.
- What rules we allow? What objects do these rules apply to?
- How do we define objects? Define syntax? Define entire logics?

Note: Notation and terminology are **VERY** important in logic.

- We want to be precise, so that we do not get contradictions!

Propositional logic

Propositions are the basic objects of study in propositional logic.

Definition: A proposition is a declarative statement that is either **true or false**, but not both.

- Examples of propositions:
  - Singapore is located in Asia. (This proposition is true.)
  - $1 + 2 = 34$. (This proposition is false.)
- Non-examples: The following statements are **not** propositions.
  - Please close the door. *(This statement is not declarative.)*
  - Who are you? *(This statement is not declarative.)*
  - There is no set X satisfying $|\mathbb{N}| < |X| < |\mathbb{R}|$. *(The continuum hypothesis is unprovable. It is neither true nor false.)*

We could give names to propositions.

- e.g. Let *Singa* be the proposition "Singapore is located in Asia."

Propositions could be **explicit** or **implicit/unspecified**.

- Symbols (e.g. *Singa*) used to explicitly represent specific propositions are called propositional constants. Symbols used to represent implicit unspecified/non-specific propositions are called propositional variables.
  - Analogous to numerical values, which could be explicit constants (e.g. 2.3, π, etc.) or unspecified constants (e.g. constant $k \in \mathbb{Z}$).

Truth values

Note: "true" and "false" are called truth values or logical values.

- The truth value representing "true" is formally denoted by $\top$.
  - Both this truth value and the symbol $\top$ are called verum.
  - Other common notation for this truth value: `True`, T, 1.
- The truth value representing "false" is formally denoted by $\bot$.
  - Both this truth value and the symbol $\bot$ are called falsum.
  - Other common notation for this truth value: `False`, F, 0.

Note: A declarative statement becomes a proposition after we assign a (fixed) truth value ($\top$ or $\bot$) to it.

Propositional logic is an example of a **two-valued logic**.

- A two-valued logic is a logic with only two truth values.
- There are other many-valued logics, e.g. three-valued logic has three truth values representing "true", "false" and "maybe".
- In this course, we will work with logical arguments based only on two-valued logic. (Statements we prove will always be either true or false.)

Operations on propositions

Suppose p and q are propositions.

- **Disjunction**: $p \vee q$ is the proposition representing "p or q".
  - If p or q is true, then $p \vee q$ is true.
- **Conjunction**: $p \wedge q$ is the proposition representing "p and q".
  - For $p \wedge q$ to be true, both p and q must be true.
- **Negation**: $\neg p$ is the proposition representing "not p".
  - If p is true, then $\neg p$ is false. If p is false, then $\neg p$ is true.
- **Implication**: $p \rightarrow q$ is the proposition representing "if p then q".
  - This proposition means "q is true on the condition that p holds."
  - $\rightarrow$ is also called material implication or material conditional.
  - p is called the premise or hypothesis or antecedent of $p \rightarrow q$.
  - q is called the conclusion or consequent of $p \rightarrow q$.
  - The proposition $p \rightarrow q$ is also called a conditional statement.
- **Biconditional**: $p \leftrightarrow q$ is the proposition $(p \rightarrow q) \wedge (q \rightarrow p)$.
  - It is the proposition that p is true **if and only if** q is true.
  - **Question:** Why is $p \leftrightarrow q$ the same as $(p \wedge q) \vee (\neg p \wedge \neg q)$?
- **Exclusive or**: $p \oplus q$ is the proposition "p or q, but not both".
  - i.e. $p \oplus q$ has the same meaning as $(p \vee q) \wedge \neg(p \wedge q)$.
  - $p \oplus q$ is also commonly denoted by $p \veebar q$.

These operations on propositions are known as **logical connectives**.

Example from textbook for implication $p \to q$

Let p, q be the propositions "I am elected" and "Taxes will be lowered".

- $p \to q$ means "If I am elected, then taxes will be lowered".
- **Recall:** $p \to q$ means "q is true on the condition that p holds."

Important note: $p \to q$ is a proposition by itself.

- This proposition is NOT p. This proposition is NOT q.

We can determine the truth value of $p \to q$ via a **truth table**:

p	q	$p \to q$	Think of $(p \to q)$ as a **promise** made by a politician John.
$\top$	$\top$	$\top$	John is elected, taxes are lowered; promise is kept.
$\top$	$\bot$	$\bot$	John is elected, taxes are NOT lowered; promise is broken.
$\bot$	$\top$	$\top$	John is NOT elected, taxes are lowered; promise is kept.
$\bot$	$\bot$	$\top$	John is NOT elected, taxes are NOT lowered; not John's fault.

Note: If John is not elected, then promise is technically automatically satisfied whether or not taxes are lowered, since the condition of John's promise (getting elected) is not satisfied.

Consequence: $p \to q$ is true, except when p is true and q is false.

More on logical connectives

Suppose p and q are propositions.

- The operations denoted by $\vee$, $\wedge$, $\neg$, $\rightarrow$, $\leftrightarrow$, $\oplus$ (or $\veebar$) are some examples of **logical connectives**.

Definition: Let $k \in \mathbb{N}$ and let $(p_1, \dots, p_k)$ be a k-tuple of propositions. In propositional logic, a k-ary logical connective is an assignment of a truth value to each of the 2^k possible k-tuples of truth values.

- $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\oplus$ are 2-ary logical connectives.
  - Intuitively, each of them "connects" two propositions and yields a truth value based on the truth values of the two propositions.
- $\neg$ is a 1-ary logial connective.
  - Intuitively, $\neg$ "connects" to just a single proposition.
- The two possible truth values $\top$, $\bot$ are 0-ary logical connectives.
- A logical connective is a k-ary logical connective for some $k \in \mathbb{N}$.

Note: New propositions can be built from existing propositions using logical connectives. These are called compound propositions.

- We can iteratively decompose a compound proposition into "shorter" propositions, until we get atomic propositions.

Logical connectives in programming

Pseudocode:

```
function FUNC($x$)
    Require: $x \in \mathbb{Z}$.
  if ($x < 5$ and $x > 0$) or $x = 10$ then
      $x \leftarrow x + 7$
```

Python code:

```
def func($x$):
    # input $x$ must be an integer.
    if ($x < 5$ and $x > 0$) or $x == 10$:
        $x = x + 7$
```

Logical connectives are found in programming languages.

- The if-condition is $((x < 5) \wedge (x > 0)) \vee (x = 10)$ or equivalently, $(p \wedge q) \vee r$, where p, q, r are propositions "$x < 5$", "$x > 0$", "$x = 10$".
- $(p \wedge q) \vee r$ is a proposition itself, so by definition, it has a truth value (true or false).

Note: In Python (and in many languages), the expression "$x == 10$" is treated as a proposition (i.e. a statement with value true/false).

- The value after "if" is expected to have Boolean type, i.e. a data type with two possible values representing truth values.
- If this value is a numerical value, then the numerical value 0 is interpreted as `False`, and all other numerical values (e.g. 1, 2.34, etc.) are interpreted as `True`.

Logical equivalence

Let p and q be propositions.

- **Remember:** $p \to q$ and $p \leftrightarrow q$ are propositions.
  - Propositions can be true or false.
- If the proposition $p \to q$ is true, then we write $p \Rightarrow q$.
  - $p \Rightarrow q$ is a proposition. It is the proposition "$p \to q$ is true".
    - It is a proposition because it is a declarative sentence.
- If the proposition $p \leftrightarrow q$ is true, then we write $p \Leftrightarrow q$.
  - $p \Leftrightarrow q$ means exactly the same as $(p \Rightarrow q) \wedge (q \Rightarrow p)$.

Definition: Let p be a compound proposition formed from atomic propositions $p_1, \ldots, p_k$ (possibly used repeatedly).

- If p is always true for all 2^k possible k-tuples of truth values, then p is called a tautology.
- If p is always false for all 2^k possible k-tuples of truth values, then p is called a contradiction.

Definition: If $p \leftrightarrow q$ is a tautology, then we say that p and q are logically equivalent, and we write $p \equiv q$.

- **Note:** p and q could be compound propositions.
- **Note:** $p \equiv q$ is not the same as $p \Leftrightarrow q$.
  - We could be lying when we declare that "$p \leftrightarrow q$" is true.

More on implications

Let p, q be propositions, and consider the implication $(p \to q)$.

- The converse of $(p \to q)$ is $(q \to p)$.
- The inverse of $(p \to q)$ is $(\neg p \to \neg q)$.
- The contrapositive of $(p \to q)$ is $(\neg q \to \neg p)$.

Important Facts:

1. $p \to q$ is logically equivalent to its contrapositive.
   - **Proof Idea:** Check the truth tables for $(p \to q)$ and $(\neg q \to \neg p)$.
2. $p \leftrightarrow q$ is logically equivalent to $(p \to q) \wedge (\neg p \to \neg q)$.
   - **Proof Idea:** Check the truth tables for both propositions.
   - $\big(p \to q$ and its converse are true$\big) \equiv \big(p \to q$ and its inverse are true$\big)$.
     - **Note:** Converse of $p \to q$ is NOT the same as inverse of $p \to q$.

De Morgan's laws for propositions

- $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$. Analogous to $(X \cup Y)^c = (X^c \cap Y^c)$.
- $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$. Analogous to $(X \cap Y)^c = (X^c \cup Y^c)$.

Example: "I don't like chocolate or chilli" $\equiv$ "I don't like chocolate and I don't like chilli."

Beyond propositions

Question: Is the statement "$x + y = 2$" a proposition?

- If x and y are **fixed** constants, then yes, "$x + y = 2$" is a proposition.
  - e.g. "$x + y = 2$" is true if x, y are both symbols for the constant 1.
  - e.g. "$x + y = 2$" is false if x, y are both symbols for the constant 0.
- What if x and y are variables?
  - In other words, what if x and y do not have fixed constant values?
  - If we allow x and y to range over integer values, then "$x + y = 2$" could sometimes be true (e.g. when $x = y = 1$), and sometimes be false (e.g. when $x = y = 0$).
- If x and y are variables, then "$x + y = 2$" is NOT a proposition!
  - But we would still like to reason about statements like "$x + y = 2$"!

Key Intuitive Idea: We can think of the statement "$x + y = 2$" as a function of the variables x and y.

- We could give the name $P(x, y)$ to the statement "$x + y = 2$".
- We could treat $P(x, y)$ as a function $P : X \times Y \to \{\top, \bot\}$.
  - X is the set of possible values for the first variable x.
  - Y is the set of possible values for the second variable y.
- **Examples:** $P(1, 1)$ is true, $P(4, -2)$ is true, $P(0, 0)$ is false.

From propositions to predicates

Consider a function from $\mathbb{Z} \times \mathbb{Z}$ to $\{\top, \bot\}$, such that:

- $(x, y) \mapsto \top$ if the statement "$x + y = 2$" is true.
- $(x, y) \mapsto \bot$ if the statement "$x + y = 2$" is false.

Note: We could equivalently think of our function as a binary relation:

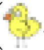
- Think of the set $P := \{(x, y) \in \mathbb{Z} \times \mathbb{Z} : \text{"}x + y = 2\text{" is true}\}$.
  - **Note:** P is a subset of $\mathbb{Z} \times \mathbb{Z}$, so P is a binary relation.
  - **Note:** $P = \{\ldots, (-1, 3), (0, 2), (1, 1), (2, 0), (3, -1), (4, -2), \ldots\}$.
  - If x is related to y by P, then we assign $P(x, y)$ the truth value $\top$.
  - If x is not related to y by P, then we assign $P(x, y)$ the truth value $\bot$.
- Thus, $P(x, y)$ is a statement whose truth value depends on the values of x, y, and the **symbol** P represents a **binary relation** from $\mathbb{Z}$ to $\mathbb{Z}$.

Definition: Let $k \in \mathbb{N}$. A k-ary predicate is a k-ary **relation** P defined on some k-tuple of sets $(X_1, \ldots, X_k)$.

- $X_1 \times \cdots \times X_k$ is called the domain of discourse for P.
- A predicate P is a k-ary predicate for some $k \in \mathbb{N}$.
- For each element $(x_1, \ldots, x_k)$ in the domain of discourse, $P(x_1, \ldots, x_k)$ is a statement that is assigned a truth value.

Examples of predicates

Example 1: A 2-ary predicate P representing "**has the color of**" is

$P = \{($, 'blue'$)$, $($  , 'yellow'$)$, $($, 'yellow'$)$, $($, 'blue'$)\}$,

where the domain of discourse for P is

$\{$, , $\} \times \{$'yellow', 'blue'$\}$

- $P($, 'blue'$)$ is a proposition that is true.
- $P($, 'yellow'$)$ is a proposition that is false.
- $P(x, y)$ is NOT a proposition, until we have fixed both x and y.
- The symbol P is not a proposition. It is a predicate.

Example 2: A 2-ary predicate P representing "**less than or equal to**" is $P = \{(1, 2), (1, 4), (1, 6), (2, 2), (2, 4), (2, 6), (5, 6)\}$, where the domain of discourse for P is $\{1, 2, 5\} \times \{2, 4, 6\}$.

- $P(1, 6)$ is a proposition that is true (since $1 \leq 6$).
- $P(5, 4)$ is a proposition that is false (since $5 \not\leq 4$).
- $P(x, y)$ NOT a proposition, until we have fixed both x and y.
- Again, the symbol P is not a proposition. It is a predicate.

More examples of predicates

Example 3: A 1-ary predicate P representing "**is a perfect square**" is $P = \{0, 1, 4, 9, 16, 25, \dots\}$, where the domain of discourse for P is $\mathbb{N}$.

- $P(81)$ is a proposition that is true (since $81 = 9^2$).
- $P(7)$ is a proposition that is false (since 7 is not a perfect square).

Example 4: A 3-ary predicate P representing "**consecutive integers**" is $P = \{\dots, (-8, -7, -6), \dots, (3, 4, 5), \dots\}$, where the domain of discourse for P is $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$.

- $P(5, 6, 7)$ is a proposition that is true.
- $P(3, 4, 9)$ is a proposition that is false.

Intuition: A predicate represents a property that elements in the domain of discourse could have.

- We have to specify the domain of discourse for each predicate, so that we are clear about the "scope" of the predicate.
- Once we fix the variables for a predicate, we get a proposition.

Note: A 0-ary predicate is by definition the same as a proposition.

- There are no variables to fix in a 0-ary predicate.

Quantifiers

Intuition: Quantification is the process of fixing variables of predicates to specific constants, so as to get propositions.

- Only after fixing variables to constants can we get truth values.

Two fundamental quantifiers: Suppose P is a 1-ary predicate.

- **Universal quantifier:** $\forall x\, P(x)$ is a proposition that declares
  - "For all x in the domain of discourse, $P(x)$ is true".
  - $\forall$ is the "for all" symbol.
- **Existential quantifier:** $\exists x\, P(x)$ is a proposition that declares
  - "There exists x in the domain of discourse such that $P(x)$ is true".
  - $\exists$ is the "exists" symbol.

Interpreting quantifiers using loops. Example: Let P be a 1-ary predicate with $\{1, \ldots, 100\}$ as its domain of discourse.

Pseudocode to compute the truth value of the proposition $\forall x\, P(x)$:

```
function FORALLP($x$)
1: for $x$ from 1 to 100 do
2:     if not $P(x)$ then
3:         return False
4: return True
```

Pseudocode to compute the truth value of the proposition $\exists x\, P(x)$:

```
function EXISTSP($x$)
1: for $x$ from 1 to 100 do
2:     if $P(x)$ then
3:         return True
4: return False
```

More on Quantification

[**Note:** Course textbook has numerous examples.]

De Morgan's law for quantifiers: Let P be a 1-ary predicate.

- $\neg\exists x\, P(x)$ is logically equivalent to $\forall x\, \neg P(x)$.
  - "There does not exist any x such that $P(x)$ is true" $\equiv$ "For all x, $P(x)$ is not true".
- $\neg\forall x\, P(x)$ is logically equivalent to $\exists x\, \neg P(x)$.
  - "It is not the case that for all x, $P(x)$ is true" $\equiv$ "There exists x such that $P(x)$ is not true"

Nested quantifiers: Let $k \in \mathbb{N}$, and let P be a k-ary predicate.

- P should be quantified by k quantifiers.
  - **Example (for $k = 3$):** $\forall x_1 \exists x_2 \forall x_3\, P(x_1, x_2, x_3)$ is a proposition that says "For all x_1, there exists some x_2 such that for all x_3, the proposition $P(x_1, x_2, x_3)$ is true."

Notation for quantifiers:

- When using quantifiers, we use ( ) appropriately for clarity, and we could restrict the domains of the quantifiers.
  - e.g. $\forall (x > 2) \forall y\, (x^2 - y^2 \geq 0)$ means we declare that "For all x in our domain satisfying $x > 2$, and for all y in our domain, the proposition "$x^2 - y^2 \geq 0$" is true."

Orders of logic

Propositional logic (also called zeroth-order logic) is a "grammar" for reasoning, consisting of the following:

- Propositions.
- Logical connectives (includes 0-ary logical connectives $\top$ and $\perp$).
- Parentheses, logical equivalence.

Predicate logic (also called first-order logic (FOL)) is a "grammar" for reasoning, consisting of the following:

- Everything in propositional logic.
- Variables (taking on values in some domain of discourse).
  - These variables are not the same as propositional variables.
- Predicates and quantifiers.

Only if you are interested: In FOL, quantifiers only quantify variables of predicates that are assumed to be elements in the domain of discourse.

- Second-order logic extends FOL by allowing quantifiers to quantify sets of elements in the domain of discourse.
- Higher-order logic extends second-order logic by allowing quantifiers to quantify iterated sets of sets of elements, etc.

Axioms and axiomatic systems

To prove theorems, we build upon simpler theorems, which in turn have proofs that are built upon even simpler theorems.

- We can iteratively deconstruct proofs of theorems into simpler and simpler theorems. However, we cannot indefinitely deconstruct proofs. We have to start somewhere!
- **Intuition:** Statements that cannot be further deconstructed are called axioms.

Definition: Axioms are statements that are declared to be true.

- A given set of axioms serves as the starting point for reasoning, from which other statements are logically derived.
- Historically, axioms are statements universally accepted to be self-evident.

Definition: An axiomatic system is a logical system for logically deriving theorems, consisting of the following:

- A logic, e.g. first-order logic.
- A finite set of axioms.

Example: Axiomatic system for arithmetic

Peano's axioms refers to the following set of nine axioms for arithmetic.

1. 0 is a natural number.
2. For every natural number x, we have $x = x$.
3. For all natural numbers x and y, if $x = y$, then $y = x$.
4. For all natural numbers x, y, z, if $x = y$ and $y = z$, then $x = z$.
5. For all x and y, if y is a natural number and $x = y$, then x is also a natural number.
6. For every natural number n, the successor of n is also a natural number.
7. For all natural numbers m and n, we have $m = n$ if and only if the successor of m equals the successor of n.
8. There is no natural number whose successor is 0.
9. (Axiom of induction) If S is a set such that $0 \in S$, and for every natural number n, we have that $n \in S$ implies the successor of n is in S, then S must contain all natural numbers.

Definition: Peano arithmetic is the axiomatic system for arithmetic, consisting of Peano's axioms and second-order logic.

Consistency versus completeness

Definition: An axiomatic system is consistent if its axioms does not lead to a logical contradiction.

- Consistency means it is not possible to prove (using the axioms) that a statement and its negation are both true.

Definition: An axiomatic system is complete if every true statement expressible using the underlying logic (e.g. second-order logic) can be proven from the given axioms.

Gödel's incompleteness theorems: (Gödel, 1931)

- (**Gödel's first incompleteness theorem**) For any axiomatic system for arithmetic that have Peano's axioms as true statements, if the axiomatic system is consistent, then it must be incomplete.
- (**Gödel's second incompleteness theorem**) For any consistent axiomatic system for arithmetic that contains Peano's axioms as true statements, it is not possible (under "general" assumptions) to prove the consistency of the axiomatic system from its axioms.

Consequence: Not every true mathematical statement is provable!

Summary

- Tuples, Cartesian products, set relations
- Propositional logic, operations on propositions
- Predicates, quantifiers, logical equivalence
- Axioms and axiomatic systems, consistency