Propositional, Predicate Logics and Set

Getinet Yilma, Ravindra Babu

- Propositional logic
- Predicate Logic
- Set Theory

Propositional logic

- Introduction to Propositional logic
- Proposition Operators
- Proposition Operators and Truth Table
- Result terminology

Propositional Logic

- Any ‘formal system’ can be considered a logic if it has
  - a well-defined syntax;
  - a well-defined semantics; and
  - a well-defined proof-theory/method/procedure.
- The **syntax** of a logic defines the syntactically acceptable objects of the language, which are properly called well-formed formulae (wff).
- The **semantics** of a logic associates each formula with a meaning.
- The **proof theory** is concerned with manipulating formulae according to certain rules.

Propositional Logic…

- A **proposition** is a statement that can be either true or false; it must be one or the other, and it cannot be both.
  - The following are propositions:
    - **the reactor is on;**
    - **the wing-flaps are up; and**
    - **a program *P* terminates.**
  - The following are not propositions
    - are you going out somewhere?
    - and 2+3
- Propositions can be atomic (single truth value) or complex (multiple joined with connectives)

Proposition connectives

- Compound propositions combined simple ones with **Logical operators**
- A truth table for some formula p, q and operators describe the behavior of a formula under all possible interpretations.

negation	(Not)	¬
conjunction	(And)	∧
disjunction	(Or)	∨
implication	(if … then)	⇒
equivalence	(if and only if)	⇔

p	*q*	*p*∧*q*	*p*∨*q*	*p*⇒*q*	*p*⇔*q*
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

P	¬*P*
T	F
F	T

Well-formed formula

- Well-formed formulas of propositional logic p and q are obtained by using the construction rules below:
  - An atomic proposition p is a well-formed formula.
  - If p and q are well-formed formulas, then so are $\neg$p, $p \wedge q$, $p \vee q$, $p \Rightarrow q$, (p) and $p \Leftrightarrow q$.

Example

- Nebil and Yasin are cousins if their fathers are brothers
  - P: Nebil and Yasin are Cousins
  - Q: Their fathers are brothers
  - P=>Q

A polygon is a triangle if and only if A polygon has exactly 3 sides

- P: A polygon is a triangle.
- Q: A polygon has exactly 3 sides
- P: x + 2 = 7 Q: x = 5
- P⇔Q

- **Tautology**: A compound proposition which is always true P or ~P is always true
- **Contradiction**: A compound proposition which is always false
- **Satisfiable (consistent)** happen when at least one (1) truth value brings the TRUE result.

Examples

- Logically Equivalent If P⇔Q is always true then P and Q are said to be logically equivalent.
- We write P≡Q e.g. ~$(P \vee Q)$ ≡ ~P^~Q

Verify the following logical equivalences

1. p∨(q∧(¬ q)) ≡ p

2. ¬(p ∧(¬ q)) ≡ (¬ p) ∨ q

Verify truth tables to check whether the following are tautologies, contradictions or neither.

p ∨ ¬(p ∨ q)

q ∨ ¬(p ∧(¬ p))

p ∨ ¬(p ∧ q)

- Can an atomic statement be a tautology or a contradiction? Explain

- $p \text{ or } true \equiv true$
- $p \text{ or } false \equiv p$
- $p \text{ and } true \equiv p$
- $p \text{ and } false \equiv false$
- $P \Rightarrow true \equiv true$
- $p \Rightarrow false \equiv not\ p$
- $ture \Rightarrow p \equiv p$
- $false \Rightarrow p \equiv true$
- $p \text{ or } p \equiv p$
- $p \text{ and } p \equiv p$
- $not\ not\ p \equiv p$
- $p \text{ or } not\ p \equiv true$
- $p \text{ and } not\ p \equiv false$

Propositional logic: limitations

- **Propositional logic:** the world is described in terms of elementary propositions and their logical combinations.
  - **A program *P* terminates**
    - **Program refers to object, terminates refers properties**
- Objects and properties are hidden in the statement, it is not possible to reason about them
- **Statements that hold for many objects must be enumerated**
- **Example:** –
- John is a SE Student → John has passed SEng3203
- Ann is a SE Student → Ann has passed SEng3203
- Ken is a SE Student → Ken has passed SEng3203
  - **Replacing** John, Ann and Ken by a variable x
  - x is a SE Student → x has passed SEng3203

Propositional logic: limitations

- **In Propositional logic statements that define the property of the group of objects are not quantified.**
  - All new cars must be registered.
  - Some of the light switches can be turned off.
- We can use quantifiers
- **Universal quantifier** –the property is satisfied by all members of the group

  $\forall$ new cars must be registered
- **Existential quantifier** – at least one member of the group satisfy the property

  $\exists$ the light switches can be turned off

Propositions in Formal Methods

- Logical propositions represent **assertions** about **system states**, which can be used to prove whether a system behaves as intended.
- Example, "If a user is logged in, they can access the file system." Or "If a user has permission, they can access the file system." This can be written as a logical proposition:

UserLoggedIn→AccessFileSystem or UserHasPermission → AccessFileSystem

Applications of Truth Tables

1. Propositional Logic: Truth tables are used to determine whether logical statements are valid, consistent, or contradictory.

2. Boolean Algebra: Truth tables help simplify complex Boolean expressions and validate logical equivalencies.

3. Digital Circuits: Truth tables are essential in designing and analyzing digital circuits, such as logic gates (AND, OR, NOT, etc.).

4. Theorem Proving: Truth tables can be used to verify logical theorems and prove the correctness of logical arguments.

1. Specification Validation: Truth tables allow developers and engineers to verify that system specifications. For example, in a formal specification for a safety-critical system, a truth table can validate whether safety properties hold in all system states.

2. Model Checking: In model checking, truth tables are used to evaluate logical properties of a system model, such as whether a certain state will always lead to a safe outcome.

- Truth tables can systematically explore all combinations of system states, helping verify properties like liveness ("something good will eventually happen") or safety ("nothing bad will ever happen").

3. Verification of Temporal Logic: In formal methods, temporal logics like Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) are used to reason about systems that evolve over time (e.g., concurrent or distributed systems). Truth tables can help verify properties like “eventually,” “always,” or “until,” where system states change as time progresses.

Role of Truth Tables in Formal Methods

- Example, Consider a system where a process A must wait for a resource B to become available before executing. A truth table could be used to verify that this condition holds under all possible states

A (Process waiting)	B (Resource available)	A ∧ B (Process can execute)
T	T	T
T	F	F
F	T	F
F	F	F

- This truth table shows that the process A can only execute (T) if both A is waiting (T) and B is available (T). Otherwise, the process cannot execute.

Predicate logic

Predicate logic

- Introduction predicate logic
- Quantifiers
- Application
- Correctness

Predicate Logic

- **Predicate logic**, **first-order logic** or **quantified logic** is a formal language in which propositions are expressed in terms of predicates, variables and **quantifiers**.
- **Quantifiers** express the extent to which a predicate is true over a range of elements.
- For example, imagine we have the statement: “Every person who is 21 years of age or older is able to purchase alcohol. Sarah is 21 years old.”
- what does “every person” really mean?
- As the range of possibilities is too broad, we need to apply one of two types of quantifiers to help us achieve our goal:
  1. Universal Quantifiers
  2. Existential Quantifier

Predicate Logic

- **Universal Quantifier:** proposition that a property is true for all the values of a variable
- P(x) is true for all values of x
- For all x, P(x)
- For each x, P(x)
- For every x, P(x)
- Given any x, P(x)
- To determine if it's a universal quantifier, you want to look for words like **all**, **each**, **every**, **any**.

Symbolically,

$\forall x \in D, P(x)$ " for all x, belonging to domain D, p(x) is true"

Predicate Logic

- **Existential Quantifier** is the proposition that a property is true for some value in a particular domain.
- There exists an x such that P(x)
- There exists an element x in the domain such that P(x)
- For some x, P(x)
- There is some x such that P(x)

And is symbolically denoted

$\exists x \in D$, P(x) "there exists x, belonging to domain D, such that p(x) is true

Suppose we have two predicate variables x and y, where the domain for x is F = {foxes} and y has the domain S = {snails}, where P(x,y) is "Foxes are faster than Snails."

Predicate Logic

Question:
"All foxes run faster than all snails."

Solution:

$\underbrace{For\ any\ fox\ x}_{\forall x \in F}$, $\underbrace{for\ any\ snail\ y}_{\forall y \in S}$, such that $\underbrace{x\ is\ faster\ than\ y}_{P(x,y)}$

$$\forall x \in F, \forall y \in S, P(x,y)$$

Question:
"Every fox is faster than some snail."

Solution:

$\underbrace{For\ any\ fox\ x}_{\forall x \in F}$, $\underbrace{there\ is\ a\ snail\ y}_{\exists y \in S}$ such that $\underbrace{x\ is\ faster\ than\ y}_{P(x,y)}$

$$\forall x \in F, \exists y \in S, P(x,y)$$

Question:
"There is a fox that is faster than all snails."

Solution

$\underbrace{There\ exists\ a\ fox\ x}_{\exists x \in F}$ such that $\underbrace{for\ any\ snail\ y}_{\forall y \in S}$, $\underbrace{x\ is\ faster\ than}_{P(x,y)}$

y

$$\exists x \in F, \forall y \in S, P(x,y)$$

Quantifiers In Truth Table

Statement	True	False
$\forall x \in D, P(x)$	$P(x)$ is true for every x	There is at least one x for which $P(x)$ is false
$\exists x \in D, P(x)$	There is at least one x for which $P(x)$ is true	$P(x)$ is false for every x

Predicate Logic

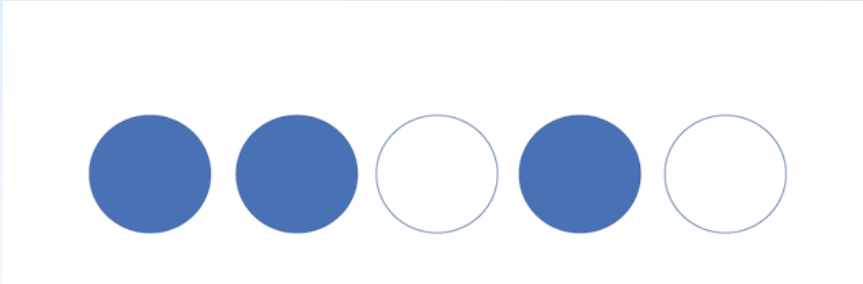
Example

1. Whoever can read is literate.
2. Dogs are not literate.
3. Some dogs are intelligent.
4. **Some who are intelligent cannot read.**

1. $\forall x\,[R(x) \Rightarrow L(x)]$
2. $\forall x\,[D(x) \Rightarrow \neg R(x)]$
1. $\exists x\,[D(x) \wedge I(x)]$
4. $\exists \mathbf{x}\,\mathbf{[I(x) \wedge \neg\, R(x)]}$

Predicate Logic

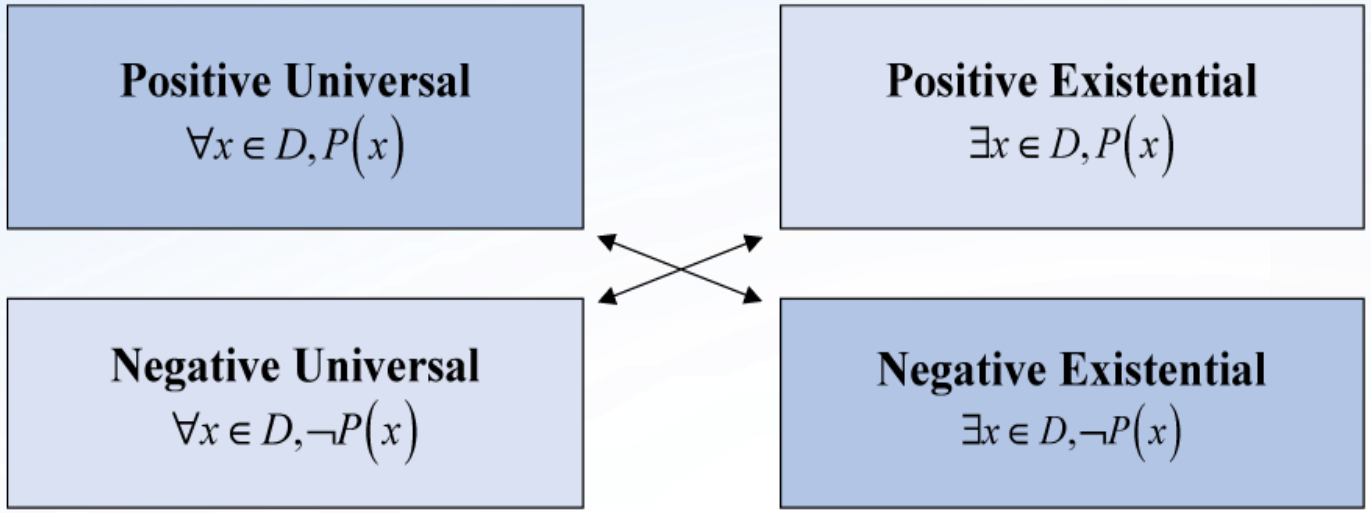
Quantifier Negation



- Using the five circles, as seen above, we make the following statement: **“Some circles are shaded.”**
- What would the negation of this statement be?
- **“Some circles are not shaded.”** is correct ?
- No! Because this statement is also trues
- Remember, a negation must have the opposite truth value.
- “All circles are not shaded.” is correct ? Yes.
- The trick is to flip the **universal quantifier** to an **existential quantifier** or vice versa, and then negate the statement, **“Some circles are shaded.”→** “**All circles are not shaded**.”
- Quantifiers give the ‘range’ and ‘universe’ over which the statement is being claimed .

Predicate Logic

The negation of quantification has the following properties:



$$\neg\left(\forall x \in D, P(x)\right) \equiv \exists x \in D, \neg P(x)$$

$$\neg\left(\exists x \in D, P(x)\right) \equiv \forall x \in D, \neg P(x)$$

Predicate Logic

- Negate the following statements with P(x) being “like homework.”

Statement
“Not all students like homework.”

Negation
“There is at least one student who does not like homework”

$$\neg\left(\forall x \in D, P(x)\right) \equiv \exists x \in D, \neg P(x)$$

Statement
“It is not the case that some students like homework.”

Negation
“All students do not like homework.”

$$\neg\left(\exists x \in D, P(x)\right) \equiv \forall x \in D, \neg P(x)$$

Predicate Logic

Negation Rules: When we negate a quantified statement, we negate all the quantifiers first, from left to right (keeping the same order), then we negative the statement.

1. $\neg[\forall x \in A, P(x)] \Leftrightarrow \exists x \in A, \neg P(x)$.
2. $\neg[\exists x \in A, P(x)] \Leftrightarrow \forall x \in A, \neg P(x)$.
3. $\neg[\forall x \in A, \exists y \in B, P(x, y)] \Leftrightarrow \exists x \in A, \forall y \in B, \neg P(x, y)$.
4. $\neg[\exists x \in A, \forall y \in B, P(x, y)] \Leftrightarrow \forall x \in A, \exists y \in B, \neg P(x, y)$.

Predicate Logic

- A **well-formed formula**, sometimes abbreviated to (wff), is obtained by composing atoms with logical connectives and quantifiers. Therefore, a well-formed formula is a predicate with the following properties

  - All atomic formulas (atoms) are well-formed formulas
  - Well-formed formulas can be combines using propositional connectives such as conjunction, disjunction, and negation
  - If x is a variable and F is a well-formed function, then both $\forall xF$ and $\exists xF$ are well-formed formulas

Example Predicate Logic

If we consider a nuclear reactor control scenario, we can adapt the example to reflect the access control logic in that context:

Consider, RodInserted(r): Predicate indicating whether control rod r is inserted.

AllowReactorControl(r): Predicate indicating whether access is granted to control the reactor with control rod r. The logic is, **∀r, (RodInserted(r) → AllowReactorControl(r))**

This statement implies that for every control rod r, if the control rod is inserted, access is granted to control the reactor with that control rod.

Reading assignments

- **Injective functions (*one-to-one*)**
- **Surjective functions (on to)**
- **Bijective functions (*one-to-one and onto*)**
- **Examples in formal methods**

Summary

- Preposition logic is describes a statement with truth value
- Operators /connectives /
- Predicate logic
- Quantifiers
- Application in formal methods

Outline

- Set Universe
- Elements
- Cardinality
- Set Operations

Set

- Sets are collection of distinct objects
- We can define a set by enumeration
  - Primary Colors = {red, yellow, blue}
  - Boolean ={true, false}
  - Evens = {…, -4, -2, 0, 2, 4, …} all these works fine for finite sets
- We can define a set by comprehension, that is, by describing a property that its elements must share.
- Two sets A and B are equal iff every member of A is a member of B and vise versa
- x ϵ S denotes “x is a member of S”
- Ø denotes the empty set

Sets : Universe

- **Notation**: { x : D | P(x) }
- Form a new set of elements drawn from domain D by including exactly the elements that satisfy predicate (i.e., Boolean function) P
- Examples:

  { x : N | x < 10} Naturals less than 10

  { x : Z | (∃ y : Z | x = 2y) } Even integers

  { x : N | x > x} Empty set of natural numbers

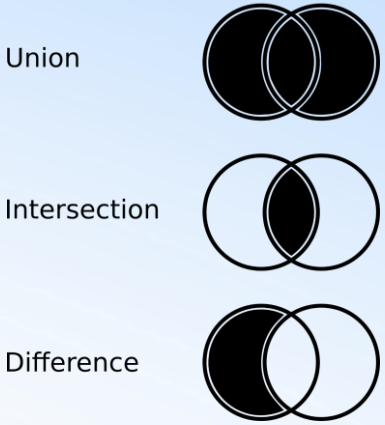
Sets: Comprehension

- Sets may be defined by **numeration** or listing their contents.
- This technique is impractical for large sets!
- Comprehension is a way of defining sets in terms of their properties or other sets.
- Example
  - $\{n : IN \mid (n \geq 0) \wedge (n \leq 3)\} = \{0, 1, 2, 3\}$
  - $\{n : Z \mid (n > 1) \wedge (n < 1)\} = \emptyset$
- **Cardinality** (#) of a finite set is the number of its elements
  - Examples: # {red, yellow, blue} = 3, # {1, 23} = 2, #∅ = 0;

$$\#(\{1, 2, 3\} \cap \{2\}) = 1$$

Set Operations

- **Union** (X, Y sets over domain D):
  - X ∪ Y ≡ {e: D | e ∈ X or e ∈ Y} → {red} ∪ {blue} = {red, blue}
- **Intersection**
  - X ∩ Y ≡ {e: D | e ∈ X and e ∈ Y}→ {red, blue} ∩ {blue, yellow} = {blue}
- **Difference**
  - X \ Y ≡ {e: D | e ∈ X and e ∉ Y}→ {red, yellow, blue} \ {blue, yellow} = {red}



- A **subset** holds elements drawn from another set
- **Proper Subsets**
- Definition: Let S and T be arbitrary sets of the same type. Then: S ⊂ T is true iff both S ⊆ T and X≠ Y. or X ⊂ Y ⇔ (X ⊆ Y) ∧ (S T)
- X ⊆ Y iff every element of X is in Y. Example {1, 2, 3} ⊆ IN
- The **power set** of set S (denoted Pow(S)) is the set of all subsets of S, i.e.,
  - Pow (S) ≡ {e | e ⊆ S}, if S has n members, then power set S has 2^n members.
  - Example: Pow({a, b, c}) = {Ø, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}}

Set operation

- Let X, Y, and Z be arbitrary sets of the same type. Then:

$$X \cup X = X$$

$$X \cap X = X$$

$$(X \cup Y) \cup Z = X \cup (Y \cup Z)$$

$$(X \cap Y) \cap Z = X \cap (Y \cap Z)$$

$$X \cup Y = Y \cup X$$

$$X \cap Y = Y \cap X$$

$$X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$$

$$X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$$

$$X \cup \emptyset = X$$

$$X \cap \emptyset = \emptyset$$

Set Equality

- Definition: Let S and T be arbitrary sets of the same type. Then S = T is true iff S and T contain precisely the same members. Example {1, 2, 3} = {2, 3, 1}

Relations and Functions

Relations

- If A and B are sets then a subset $R \subseteq A \times B$ is a relation between A and B. People often write xRy to mean $(x, y) \in R$
- A relation R between A and itself is
  - reflexive, if $\forall x \in A.(x, x) \in R$ (can always go from x to itself)
  - transitive, if $\forall x, y, z.(x, y) \in R \wedge (y, z) \in R \rightarrow (x, z) \in R$

Functions

- A relation R between A and B is functional if both
- $\forall x \in A.\exists y \in B.(x, y) \in R$ (left-totality, every x maps somewhere)
- $\forall x, y, z.(x, y) \in R \wedge (x, z) \in R \rightarrow y = z$ (right-uniqueness, only one function value)
- If f is a functional relation between A and B, we call f a function and write

$$f : A \rightarrow B.$$

Applications Set Theory

- Set theory is extensively used in relation and relational modeling
  - Relational algebra
  - Relational calculus
  - SQL query
    - Cartesian product
    - Joins
    - Set operation such as union, intersection, set difference and
    - Other compound operations
- Set theory is also used to model relational mapping (functions)

Summary

- Set theory
- Set operators- union, intersection, set difference, sub sets, proper subsets, power sets etc.
- Set theory applications in databases applications and relational mapping