CS 1010 Discrete Structures
Lecture 2:
Predicate Logic

Maria Francis

November 26, 2020

Applications: Consistency of Requirements

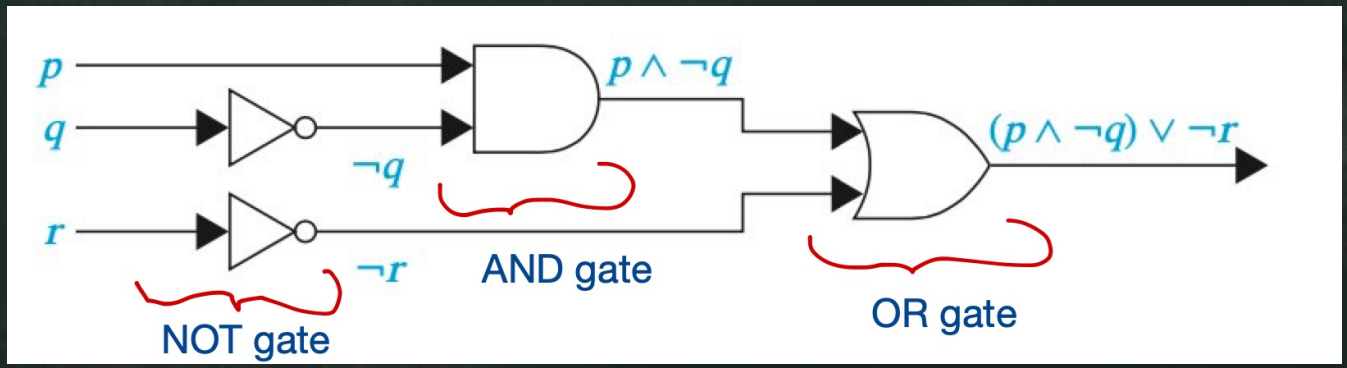
- How to specify system requirements in a consistent manner.
- Example: Are these system specifications consistent.
  1. The diagnostic message is stored in the buffer or it is retransmitted.
  2. The diagnostic message is not stored in the buffer.
  3. If the diagnostic message is stored in the buffer, then it is retransmitted.
- p: The diagnostic message is stored in the buffer.
- q: The diagnostic message is retransmitted.
  1. $p \vee q$
  2. $\neg p$
  3. $p \rightarrow q$

Applications: Consistency of Requirements

- We need an assignment of truth values such that p is false since $\neg p$ needs to be true.
- Since we want $p \vee q$ to be true q has to be true.
- We want $p \rightarrow q$ to be true and that happens only when p is false and q is true.
- System specifications are consistent because they are all true when p is false and q is true.

Other Applications

- Boolean Searches in webpage searching: Search engines support Boolean searching techniques. For eg: "IIT" $\wedge$ "Hyderabad" $\wedge$ "CSE".
- Logic/Digital Circuits: They receive input signals $p_1, p_2, \ldots, p_n$ (each taking off or on values) and produces output signals $s_1, s_2, \ldots, s_m$ (again off and on).



Predicates & Quantifiers

- Propositional logic is not enough to express the meaning of all statements in mathematics or conversational language.
- $x^2 \geq 0$ is always true for *all* $x \in \mathbb{R}$.
- $5x^2 - 7 = 0$ is true for *some* x, $x = \pm\sqrt{7/5}$.
- The need to quantify, i.e. how often the predicate is true comes up many times when you write mathematics :
  1. For all $x \in D$, $p(x)$ is true.
  2. $p(x)$ is true for at least one x in D.
- Always true : Universal Quantification
- Sometimes true: Existential Quantification
- Predicate logic/First order logic captures that : propositional logic + quantifiers.

Predicates

- The statement x is greater than 3 has two parts : variable x and predicate: "greater than 3".
- We denote the statement x is greater than 3 by $P(x)$ where P denotes the predicate part.
- $P(x)$ is also called the value of the proposition function P at x.
- *Once you assign a value to x, $P(x)$ becomes a proposition and has a truth value.*
- $P(x)$: $x > 3$. Truth values at $P(4)$ is T and at $P(2)$ is F.

Predicates

- You can have more than one variable in a predicate: $P(x_1, x_2, \ldots, x_n)$.
- You have then a n-ary predicate P whose value at the n-tuple $(x_1, x_2, \ldots, x_n)$ is $P(x_1, \ldots, x_n)$.
- Predicates can be used to establish the correctness of computer programs : when given valid input the program produces valid outputs.
- Preconditions: Statements that describe valid input.
- Postconditions: Conditions that a valid output should satisfy.

Correctness of Programs

- Consider the program that looks at the interchange of two variables x and y :
  1. `temp:=x`
  2. `x:=y`
  3. `y:=temp`
- Precondition: $P(x, y) : x = a$ and $y = b$ where a and b are the values of x and y before we run the program.
- Postcondition: $Q(x, y) : x = b$ and $y = a$.

Universal Quantifier

- A property is true for all values values of a variable in a domain (domain of discourse/universe of discourse).
- $\forall x P(x)$ denotes the universal quantification of $P(x)$.
- $P(x)$ for all values of x in the domain.
- $\forall$ is the universal quantifier.
- An element for which $P(x)$ is false is a counterexample of $\forall x P(x)$.
- For all/For every/For each/For any/For arbitrary : all ways of expressing universal quantification.
- For any can be ambiguous.
- If you can list out all the elements of the domain x_i then $\forall x P(x)$ is the conjunction of $P(x_i)$s.

Existential Quantifier

- We form a proposition that is true iff $P(x)$ is true for at least one value of x in the domain.
- $\exists$ is the existential quantifier.
- Without domain no meaning. If domain is empty then $\exists x P(x)$ is false.
- For some/For at least one/ There is : all ways of expressing existential quantification.
- If you can list out all the elements of the domain x_i then $\forall x P(x)$ is the disjunction of $P(x_i)$s.

Variations of Quantifier

- Uniqueness identifier : when we want to say there is a unique x for which $P(x)$ is true.
  - $\exists! x P(x)$ or $\exists_1$: notation for that.
  - $\exists! x(x-1)=0,\ x \in \mathbb{R}$
- Restricted Domains:
  - $\forall x < 0(x^2 > 0)$ and $\exists z > 0(z^2 = 2)$ where domain is $\mathbb{R}$.
  - $\forall x < 0(x^2 > 0)$: Square of a negative real number is positive. $\forall x(x < 0 \rightarrow x^2 > 0)$.
  - $\exists z > 0(z^2 = 2)$: There is a positive square root of 2. $\exists z(z > 0 \wedge z^2 = 2)$
  - Restricting $\forall \equiv \forall$ of a conditional statement.
  - Restricting $\exists \equiv \exists$ of a conjunction.

Binding Quantifiers

- When a quantifier is used on a variable x, that occurrence of the variable is bound.
- If a variable is not bound by a quantifier or is not set to be equal to a particular value then it is said to be free.
- The part of the expression where the quantifier is applied is called scope of the quantifier. Free variables are outside the scope of all quantifiers.
  - $\exists x(x + y = 1)$ has y a free variable and x bound by $\exists x$.
  - $\exists x(P(x) \wedge Q(x)) \vee \forall x\ R(x)$. All variables are bound. The scope of $\exists x$ is $P(x) \wedge Q(x)$ and for $\forall x$ it is $R(x)$.
  - Since scopes do not overlap we could have used different variables. But same variables for different scopes are also commonly used.

Equivalences with Quantifiers

Logically equivalence iff they have the same truth value no matter which predicates are substituted and which domain of discourse is used for the variables in the propositions.

- $\forall x\ (P(x) \wedge Q(x))$ and $\forall x\ P(x) \wedge \forall x\ Q(x)$ are logically equivalent. Verify!
- What about $\forall x\ (P(x) \rightarrow Q(x))$ and $\forall x\ P(x) \rightarrow \forall x\ Q(x)$? Are they logically equivalent?
  - No!
  - If the hypothesis is false a conditional statement is true.
  - Let us set $P(x)$: integer x is a multiple of 2 and $Q(x)$: integer x is a multiple of 4.
  - $\forall x\ P(x) \rightarrow \forall x\ Q(x)$ is true since $\forall x\ P(x)$ is false and we dont care for the conclusion.
  - But $\forall x\ (P(x) \rightarrow Q(x))$ is false!
  - Note that for some choices it works like when P and Q have same truth values.

Negating Quantifiers

- Every student has taken a course in calculus. $\forall x\ P(x)$.
- Negation where domain is students in my class: It is not the case that every student in my class has taken a course in calculus.
- Or There is at least one student in my class who has not taken a course in calculus.
- Equivalent to $\exists x\ \neg P(x)$.

$$\neg \forall x\ P(x) \equiv \exists x\ \neg P(x).$$

Negating Quantifiers

- What about $\neg\exists\ Q(x)$? Say, $Q(x)$ is x has taken a course in calculus.
- Negation: There is not a single student who has taken a course in calculus in my class.
- Every student in my class has not taken calculus. $\forall x\ \neg Q(x)$

$$\neg\exists x\ Q(x) \equiv \forall x\ \neg Q(x).$$

- The rules for negation for quantifiers are called De Morgan's laws for quantifiers.
- Examples:
  - $\neg\forall x(x^2 > x) : \equiv \exists x\ \neg(x^2 > x) \equiv \exists x\ x^2 \leq x.$
  - $\neg\exists x\ (x^2 = 2) : \equiv \forall x \neg(x^2 = 2) \equiv \forall x(x^2 \neq 2).$

Negating Quantifiers

- S.T. $\neg\forall x\ (P(x) \rightarrow Q(x)) \equiv \exists x(P(x) \wedge \neg Q(x))$.
- $\neg\forall x\ (P(x) \rightarrow Q(x)) \equiv \exists x(\neg(P(x) \rightarrow Q(x)))$ (De Morgan's)
- $\neg(P(x) \rightarrow Q(x)) \equiv P(x) \wedge \neg Q(x)$ (Implication equivalences)
- Thus we have the answer.

System Specifications using Quantifiers

- Every mail message larger than 1 MB will be compressed.
- $S(m, y)$: Mail message m is larger than y MB, m is from the domain of all messages and $y \in \mathbb{R}^+$.
- $C(m)$: Message m will be compressed.
- $\forall m\ (S(m, 1) \rightarrow C(m))$

Nested Quantifiers

- One nested quantifier in the scope of another:

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

- Whatever is inside the scope of a quantifier can be thought of as a proposition function.
- That is, $\forall x Q(x)$ where $Q(x)$ is $\exists y\ P(x, y)$, where $P(x, y)$ is $x + y = 0$.
  1. $\forall x \forall y \forall z (x + (y + z) = (x + y) + z)$ is the associative law for addition of real numbers.
  2. $\forall x \forall y\ ((x > 0) \wedge (y < 0) \rightarrow (xy < 0))$, domain: $\mathbb{R}$. For every real number x and for every real number y, if $x > 0$ and $y < 0$ then $xy < 0$. Product of a positive real number and a negative real number is always a negative real number.

Swapping Quantifiers

- Goldbach's Conjecture: Every even integer greater than 2 is the sum of two primes.
- For every even integer n greater than 2, there exist primes p and q such that $n = p + q$.
- Let *Evens* be set of even integers greater than 2 and *Primes* the set of primes.
- $\forall n \in \mathit{Evens}\ \exists p \in \mathit{Primes}\ \exists q \in \mathit{Primes}\ n = p + q$.
- Order of precedence: $\forall$ and $\exists$ have higher precedence than all logical operators of propositional calculus.

Swapping Quantifiers

- Swapping the order of quantifiers will change the meaning.
- Every child has a dream. Let set of all children be C and D the set of dreams and $H(c, d)$: Child c has dream d.
- $\exists d \in D\ \forall c \in C\ H(c, d)$: every child has the same dream.
- $\forall c \in C\ \exists d \in D\ H(c, d)$: every child has a specific dream.
- Swapping in Goldbach Conjecture:

$$\exists p \in Primes\ \exists q \in Primes, \forall n \in Evens\ n = p + q.$$

- Every even number $gneq2$ is the sum of *same* two primes. Not true!
- Verify! If $\exists y\ \forall x\ P(x, y)$ true then $\forall x\ \exists y\ P(x, y)$ is true. But not vice-versa!

Quantification as Loops

- Quantifiers can be viewed in terms of nested (possibly infinite) loops.
- $\forall x \forall y\ P(x, y)$: we loop through all values for x and for each x you loop through all the values of y.
- $\forall x\ \exists y\ P(x, y)$ is true if we loop through all values of x and for each x we loop through all the values of y and we find one value for which $P(x, y)$ is true.
- Similarly for $\exists x\ \forall y\ P(x, y)$ and $\exists x\ \exists y\ P(x, y)$

Negating Nested Quantifiers

- Negation of $\forall x\ \exists y\ (xy = 1)$:
  - $\equiv \exists x \neg \exists y\ (xy = 1)$
  - $\equiv \exists x \forall y\ \neg(xy = 1)$.
  - $\equiv \exists x \forall y\ xy \neq 1$.

Arguments

- Argument: Sequence of statements that end with a conclusion.
- Valid: Conclusion of the argument should follow truth from the preceding statements/premises.
- An argument is valid iff it is impossible for all premises to be true and the conclusion to be false.
- Proofs that we will soon see are valid arguments that establish the truth of the statements.

Argument Form

- For now, we confine the discussion to compound propositions and no other statements.
- We introduce argument form where the propositions are replaced by propositional variables.
- If $(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \rightarrow q$ is a tautology then we can say that the argument with premises $p_1, p_2, \ldots, p_n$ and conclusion q is valid.
- We use rules of inference to get to valid arguments. We use a lot of this in proofs in computer sc and mathematics without explicitly stating it.

Rules of Inference

Modus Ponens

$$\begin{array}{ll} & p \\ & \underline{p \to q} \\ \therefore & q \end{array}$$

i.e. $(p \wedge (p \to q)) \to q$ is a tautology.

Modus Tollens

$$\begin{array}{ll} & \neg q \\ & \underline{p \to q} \\ \therefore & \neg p \end{array}$$

Rules of Inference

Hypothetical Syllogism

$$\begin{array}{ll} & p \to q \\ & \underline{q \to r} \\ \therefore & p \to r \end{array}$$

Disjunctive Syllogism

$$\begin{array}{ll} & p \vee q \\ & \underline{\neg p} \\ \therefore & q \end{array}$$

Rules of Inference

Addition

$$\begin{array}{ll} & \underline{p} \\ \therefore & p \vee q \end{array}$$

Simplication

$$\begin{array}{ll} & \underline{p \wedge q} \\ \therefore & p \end{array}$$

Rules of Inference

Conjunction

$$\begin{array}{ll} & p \\ & \underline{q} \\ \therefore & p \wedge q \end{array}$$

Resolution

$$\begin{array}{lll} & p \vee q & \\ & \underline{\neg p \vee r} & \\ \therefore & q \vee r & (\textit{resolvent}) \end{array}$$

Used in computer programs to automate proving theorems.

Examples

- p : It is below freezing now. Therefore it is either below freezing or raining now.
  - Addition rule!

$$\begin{array}{ll} & \underline{p} \\ \therefore & p \vee q \end{array}$$

Examples

- p: It is sunny this afternoon, q: It is colder than yesterday, r: We will go swimming, s: We will take a boat trip, t: We will go home by sunset
- Premises: $\neg p \wedge q$, $r \rightarrow p$, $\neg r \rightarrow s$, $s \rightarrow t$.
- Conclusion : t Can we have a valid argument with these premises and conclusion t?
  1. $\neg p \wedge q$ Premise
  2. $\neg p$ From (1)
  3. $r \rightarrow p$ Premise
  4. $\neg r$ Modus Tollens
  5. $\neg r \rightarrow s$ Premise
  6. s Modus Ponens
  7. and the premise $s \rightarrow t$ gives us t.

Other Syntax

- In another convention the set of premises are called axioms and using a set of rules of inference you conclude about a new formula.
- Notation: $\varphi_1, \varphi_2, \cdots, \varphi_n \vdash \psi$ where $\vdash$ means infer and ψ is the conclusion.
- A proof or derivation in an axiom system where each proposition/formula is either an axiom or follows the previous propositions via an inference rule.
- ψ can be inferred from an axiom system, i.e. $\vdash \psi$, if there is a derivation from the premises that ends with ψ.

Completeness and Soundness

- Derivation and inference has nothing to do with truth and validity.
  - Start with false axioms or illogical rules we get an invalid formula. Fallacy of affirming the conclusion.
  - Or, if we start with incomplete axioms or we miss some rules we wont be able to derive valid formulas.
- We want an axiom system to be complete and sound:
  - Completeness: All valid statements can be derived.
  - Soundness: Only valid systems can be derived.

Rules for Quantifiers

Universal Instantiation

$$\frac{\forall x P(x)}{\therefore \quad P(c)}$$

Universal Generalization

$$\frac{P(c) \text{for an arbitrary } c}{\therefore \quad \forall x P(x)}$$

Rules for Quantifiers

Existential Instantiation

$$\frac{\exists x P(x)}{\therefore \quad P(c) \text{ for some element} c}$$

Existential Generalization

$$\frac{P(c) \text{ for some element } c}{\therefore \quad \exists x P(x)}$$

Example

- Divya from this class knows how to write programs in Python. All Python coders will get a data science job. Therefore, someone in this class will get a data science job.
- $p(x)$: x is in this class, $q(x)$: x knows how to write Python programs, $r(x)$: x will get a data science job
- Premises: $p(\text{Divya})$, $q(\text{Divya})$, $\forall x(q(x) \rightarrow r(x))$.
- Conclusion : $\exists x(p(x) \wedge r(x))$

Example

1. $\forall x(q(x) \rightarrow r(x))$ (Premise)
2. $q(\text{Divya}) \rightarrow r(\text{Divya}))$ (Universal Instantiation)
3. $q(\text{Divya})$ (Premise)
4. $r(\text{Divya}))$ (Modus Ponens)
5. $p(\text{Divya})$ (Premise)
6. $r(\text{Divya})) \wedge p(\text{Divya})$ (Conjunction)
7. $\exists x(p(x) \wedge r(x))$ (Existential Generalization)

Combining rules for Quantifiers & Propositions

- Universal instantiation + modus ponens : universal modus ponens. Very commonly used.
- What does that mean?
  - $\forall x\ (P(x) \to Q(x))$ is true and $P(a)$ is true for some a then $Q(a)$ is true.
  - Universal instantiation says: $P(a) \to Q(a)$ is true and modus ponens says $Q(a)$ must also be true.
- Similarly, universal modus tollens.