CS 245

- Natural Language

Propositional Logic

Logic is the systematic study of the principles of reasoning and inference. - Used for modeling computer hardware, software, and embedded systems. - Designing systems that can apply reason and inferences (artificial intelligence)

A **proposition** is a declarative sentence that is either **true** or **false**.

English to Propositional Logic

$\neg p$: not p, it is not the case of p

$p \wedge q$: p and q, p but q, not only p but q, p while q, p despite q, p yet q, p although q

$p \vee q$: p or q, p and/or q, p unless q

$p \implies q$: if p then q, q if p, p only if q, q when p, p is sufficient for q, q is necessary for p

$p \iff q$: p if and only if q, p is equivalent to q, p exactly if q, p is necessary and sufficient for q

A logic is formalized for syntax, semantics, and proof procedures.

Syntax

Atomic propositions are combined into compound propositions, and are analyzed as a set of interrelated propositions.

Propositions are represented by formulas which contains a sequence of symbols. An **expression** is a finite sequence of symbols. - Propositional variables - Denoted with lowercase Roman letters - Connectives - Logic operators (NOT, AND, OR) - Punctuation - Brackets

Binary connectives (AND, OR, IMPLIES) apply to two things. **Unary** connectives (NOT) are only applied to one.

A **formula** can be built from a set of proposition variables. A formula can be atom, negation, conjunction, disjunction, implication. It should only take on the type of the outermost type.

Example: $(p \wedge q) \implies r$ is an implication.

Semantics

The **semantics** of a logic describes how to interpret the well-formed formulas of the logic.

A **truth valuation** is a function with the set of all proposition symbols as domain and assigns a truth value (T,F) to each propositional variable.

For example $t(p) = p^t = T$

Everything can be built from the nand symbol. (see computer boolean)

To prove theorems, mathematical induction is used.

Principle of Mathematical Induction:

Establish that the base case has a property P. Show that if the property holds for the base case, the next number also holds the property. Conclude that the property holds for all $n \in N, n \geq BC$

Theorem: Let R be a property, suppose that

1. for each atomic formula p, we have $R(p)$
2. For each formula α, if $R(\alpha)$ then $R((\neg\alpha))$
3. For each pair of formulas α and n, and each connective $*$, if $R(\alpha)$ and $R(n)$, then $R((\alpha * n))$.

This is called **structural induction**.

Something happened here

Working with Formulas

A formula α is a **tautology** if and only if for every truth valuation t, $\alpha^t = T$

A formula α is a **contradiction** if and only if for every truth valuation t, $\alpha^t = F$. Example: $p \wedge \neg p$

A formula α is **satisfiable** if and only if there is some truth valuation t, such that $\alpha^t = T$

Note: A formula is satisfiable if and only if it is not a contradiction.

A **valuation tree** is a shortcut to analyzing what would happen if the entire truth table was filled out.

Two formulas are **equivalent**, $\alpha \equiv \beta$ if $\alpha^t = \beta^t$ for every valuation t.

Lemma: Suppose that $\alpha \equiv \beta$. Then for any formula γ, and any connective $*$, the formula $\alpha * \gamma$ and $\beta * \gamma$ are equivalent.

$$\alpha * \gamma \equiv \beta * \gamma *$$

Example: Prove $(p \wedge q) \vee (q \wedge r) \equiv (q \wedge (p \vee r))$

Proof: $= (p \wedge q) \vee (q \wedge r)$

$= (q \wedge p) \vee (q \wedge r)$

$= (q \wedge (p \vee r))$

Simplify: $p \vee (p \wedge q)$

Solution:

$= (p \wedge T) \vee (p \wedge q)$

$= p \wedge (T \vee q)$

$= p$

Class

Prove $\neg(((\neg p \vee q) \vee r) \wedge \neg r \wedge p) \vee q$

Example: Prove or disprove $p \equiv p \wedge (q \implies p)$

$p \wedge (q \implies p)$

$p \wedge (q \vee \neg p)$ Implication

$p \wedge (\neg p \vee q)$ Commutativity

p Simplification II

LHS=RHS

Example: Prove or disprove $p \wedge (\neg(\neg q \wedge \neg p) \vee p) \equiv q$

Solution: Find an counter example. Select $t(p) = F, t(q) = T$. LHS = F, RHS = T

Alternatively, simplify first if a value cannot be found by inspection.

$p \wedge ((q \vee p) \vee p)$

$p \wedge (q \vee (p \vee p))$

$p \wedge (q \vee p)$

$p \wedge (p \vee q)$

p

Logical Consequence

Let σ be a set of formulas and let ϕ be a formula. We say that

- ϕ is a logical consequence of σ or
- σ semantically entails ϕ, or
- in symbols, $\sigma| = \phi$

if and and only if for any truth value t

$$\sigma^t = T \implies \phi^t = T$$

Example:

$$\{(p \implies q), q\}| = q$$

Solution: Whenever $\{(p \implies q), q\}$, q must be true. This can be shown from drawing a truth table.

Example: $\{p \implies q, q\}| \neq p$

Solution: When $p = F, q = T$, and the proposition does not hold.

Example:

$$\{(p \implies q), (q \implies r)\}| = (r \implies q)$$

Solution: Draw huge truth table. $p = F, q = F, r = T$ has $\sigma^t = T$ and $\phi^t = F$

Note: If σ is a contradiction, $\sigma| = \phi$ will always be true.

Dealing with Code

If statement fragments can be simplified by drawing a truth table and taking the simplest expressions. The **else** block should be the largest block to avoid writing extra code.

Formulas $\alpha \implies \beta$ and $\neg\alpha \vee \beta$ are equivalent. Thus $\implies$ is said to be **definable** in terms of $\neg$ and $\vee$.

There are sixteen possible binary connectives. (16 because there are 16 way to permutate T and F over the truth table containing 4 slots. Two are nullary (ignores the value they connect), and four others are unary (ignore one value but not the other).

A set of connectives is **adequate** iff any n-ary $(n \geq 1)$ connective can be defined in terms of ones in the set.

$(\vee, \wedge, \neg)$ is an adequate set of connectives.

$(\wedge, \neg)$,$(\vee, \neg)$,$(\implies, \neg)$ are adequate sets of connectives.

Section

A **proof** is a formal demonstration that a statement is true. I

- It must be mechanically checkable. Intuition and insight should not be necessary.
- Any computer could verify its correctness

A proof is generally syntactic, rather than semantic. Permits mechanical checking. Rules chosen for semantic reasons.

A proof consists of a list of formulas. Assumption $\rightarrow$ inference $\rightarrow$ inference rules $\rightarrow$ conclusion.

A set of inference rules defines a **proof system**.

We notate "there is a proof with assumptions $\sum$ and conclusion ϕ" by $\sum \vdash \phi$

An inference rule is written as

$$\frac{a_1 \qquad a_2 \cdots a_i}{\beta}$$

This means, suppose each formula in α already appears in the proof (either assumed or previously inferred), then one may infer the formula β

Example $\frac{\alpha\,\beta}{\alpha \wedge \beta}$

Approaches to Proofs

Direct proof. Establish $\sum \vDash \phi$. Give a proof with $\alpha_1, \alpha_2, \ldots, \alpha_n$ as assumptions and obtain ϕ as conclusion.

Refutations (indirect proofs, proof by contradiction)

Establish $\sum \vDash \phi$. Assume $\neg\phi$ and $\alpha_1, \alpha_2, \ldots, \alpha_n$ and obtain a contradiction as conclusion. This works because any valuation that makes $\sum$ true must make $\neg\phi$ false and make ϕ true.

Resolution is a refutation system with the following influence rule.

$\frac{a \vee p \ \neg p \wedge b}{a \vee b}$

for any variable p and formula a and b.

Consider the following as special cases

Unit Resolution $\frac{a \lor p \qquad \neg p}{a}$

Contradiction: $\frac{p \qquad \neg p}{\text{Contradiction}}$

Resolution

To prove $\{p, q\}_{\text{res}} p \land q$, derive a contradiction from the assumptions $\{p, q, \neg(p \land q)\}$

Conjunctive normal form

- A **literal** is a (propositional) variable or the negation of a variable
- A **clause** is a disjunction of literals (cannot use and)
- A formula is in **conjunctive normal form** if it is a conjunction of clauses.

A formula is in CNF if and only if

- its only connectives are $\neg, \lor, \land$
- $\neg$ applies only to variables
- $\lor$ applies only to subformulas with no occurrence of $\land$.

To prove ϕ from $\sum$ via resolution refutation

- Convert each formula in $\sum$ to CNF
- Convert $\neg\phi$ to CNF
- Split the CNF formulas as the $\land$s, yielding a set of clauses
- From the resulting set of clauses, keep applying the resolution inference rule until
  - The empty clause (contradiction) results (ϕ is a theorem)
  - The rule cannot be applied to give a new formula (ϕ is not a theorem)

TODO: Resolution (2 classes)

Predicate Logic | First Order Logic

Propositional logic dealt with logical forms of compound propositions that talked about all objects in a set. Predicate logic deals with *some* object, without enumerating all objects in a set.

Example: Not all birds can fly

$$\neg(\forall x \cdot (B(x) \implies F(x)))$$

Ingredients

First order logic is expressed with the following ingredients

- A non-empty domain of objects (individuals, sets)
- Names of individuals (also called constants)
- Variable denoting generic objects
- Relation (eg: equal, younger)
- Function (eg: +, mother of)
- Quantifiers
- Propositional connectives

A k-ary relation is a set of k-tuples of domain elements.

Example: Binary relation less-than, over a domain D, is represented by the set

$$\{(x, y) \in D^2 \mid x < y\}$$

Variables let us refer to an object without specifying which particular object it is.

In first order logic, functions may be used as arguments to another function.

Example: Every child is younger than its mother.

$$\forall x \cdot \forall y \cdot ((C(x) \wedge M(y, x)) \implies Y(x, y))$$

But this allows x to have several mothers as long as $M(y, x)$ is satisfied.

The above example can be rewritten as

$$\forall x \cdot (C(x) \implies Y(x, m(x)))$$

Syntax

Expressions that have a truth value are **formulas**

Expressions that refer to an object of the domain are called **terms**.

An **atomic formula** is an expression in the form $P(t_1, \ldots, t_n)$ such that P is an n-ary function and each t_i is a term.

$\forall x \cdot \alpha$ is a formula if α is a formula, and the formula α is the scope of the quantifier.

Interpretation

Fix a set C of constant symbols, function symbols, and relation symbols.

An interpretation M for the set C consists of

- A non empty set $\text{dom}(M)$, called the domain of M
- For each constant c, a member c^M of $\text{dom}(M)$
- For each function symbol $f^{(i)}$, an iary function f^M
- For each relation symbol $R^{(i)}$, an iary function relation R^M

An interpretation is also called a **model**.

With an interpretation M, for each term t containing no variables, the value of t under interpretation M, denoted t^M is

- If t is a constant c, t^M is c^M
- If t is a function, the value t^M is $f^M(t_1^M, \ldots, t_n^M$

The value of a term must always be a member of domain M

With an interpretation M, for each formula α containing no variables, the value of α under interpretation M, denoted α^M is

- If α is $R(t_1, \ldots, t_n)$, then a^M is T if $t_1^M, \ldots, t_n^M \in R^M$. Else it is F
- If α is $\neg\beta$ or $\beta \star \gamma$, then α^M is determined the same way as for propositional logic.

Example: Consider an interpretation M with

- Domain: N, the natural numbers
- 0^M: zero
- f^M: successor
- E^M: is even

$f(0)^M = 1$, $E(f(0))^M = F$, $E(f(f(0)))^M = T$

NOTE: There is NO default meaning for relation, function, or constant symbols. Functions must be defined at every point in the domain.

An occurrence of a variable in a formula is **bound** if it lies in the scope of some quantifier of the variable, otherwise it is **free**.

Example: In the formula $P(x) \wedge \forall x \cdot Q(x)$, first x is free and last x is bound.

Substitution

The notation $\alpha[t/x]$ for a variable x, term t, and formula α denotes the formula obtained from α by replacing each free occurrence of x with t

NOTE: Substitution only affects free variables, not bound occurrences.

Semantics

In propositional logic, semantics was described in terms of valuations to propositional atoms. FOL includes more ingredients and the semantics must account for them too.

An **environment** is a function that assigns a value in the domain to each variable.

Example: With domain N, we might have environment θ_1, given by $\theta_1(x) = 9$ and $\theta_1(y) = 2$.

Creating a formula

To create a formula, we must specify the intepretation (includes domain, definition of symbols, and constants) and an environment that maps variables

Example: Let dom$\{J\}$ be the collection of all words over the alphabet $\{a, b\}$

$0^J = a$, s^J is appending a to the end of the string, and $+$ is a concatenation.

$\theta(x) = aba$

Evaluating $(s(s(0) + s(x)))^{(J,\theta)}$ gives $aaabaaa$

Quantified Formulas

For any environment $]theta$ and domain element d, the environment θ with x reassigned to d is denoted as $\theta[x \to d]$

In this case, x is substituted with d for every quantified occurrence.

Cannot directly say a statement is true for every $x \in dom(M)$, must say $\alpha^{(M,[x \to d])}$ for every $d \in dom(M)$

NOTE: The values of $(\forall x \cdot \alpha)^{(M,\theta)}$ and $(\exists x \cdot \alpha)^{(M,\theta)}$ do not depend on the value of $\theta(x)$ as it only matters for free occurrences.

If $M \vDash_\theta \alpha$ for every θ, then M **satisfies** α, denoted $M \vDash \alpha$

Validity and Satisfiability

A first order formula can be either valid, satisfiable, and unsatisfiable. There are no tautologies in predicate logic.

Suppose σ is a set of formulas and α is a formula. α is the **logical consequence** of $\sigma \vDash \alpha$ iff for any interpretation M and environment θ, we have $M \vDash_\theta \sigma$ implies $M \vDash_\theta \alpha$

Example: Show that $\forall x \cdot \neg\gamma \vDash \neg\exists x \cdot \gamma$

$M \vDash_\theta \forall x \cdot \neg\gamma$. This means for every $a \in dom(M)$, $M \vDash_{\theta[x \to a]} \neg\gamma$

By definition, that means for $a \in dom(M)$, $M \nvDash_{\theta[x \to a]} \gamma$

This means there is no $a \in dom(M)$ that satisfies $M \vDash_{\theta[x \to a]} \gamma$

This is the RHS of the equation as required.

Proofs in FOL

$\forall$ elimination: $\frac{\forall x \cdot \alpha}{\alpha[t/x]}$

$\exists$ introduction: $\frac{\alpha[t/x]}{\exists x \cdot \alpha}$

A variable is **fresh** in a subproof if it occurs nowhere outside the box of the subproof.

If you start with y fresh and can conclude $\alpha[y/x]$, you can conclude $\forall x \cdot \alpha$

If you start with $\exists x \cdot a$ and when you introduce u fresh you can conclude β, then you can actually conclude β

Definition: First Order Logic with Equality is First-Order Logic with the restriction that the symbol = must be interpreted as equality on the domain

$$(=)^L = \{(d, d) | d \in dom(L)\}$$

If $t = t$, you can do an equals introduction

If $t_1 = t_2$ and $\alpha[t_1/x]$ then you can conclude $\alpha[t_2/x]$

An **axiom** is a premise that is always taken, it need not be listed explicitly.

EQ1: $\forall x \cdot x = x$ is n axiom.

EQ2: For each formula α and variable z

$\forall x \cdot \forall y \cdot (x = y \implies (\alpha[x/z] \implies \alpha[y/z]))$

These axioms then imply symmetry and transitivity.

Arithmetic, Data Structures, and Programs

Relations on Lists

If x is the empty list

$\forall x \cdot \forall y \cdot (x = e) \implies \neg \text{ First}(x, y)$

$\forall y \cdot \neg \text{ First}(x, y)$

x is not the empty list

$\forall x \cdot \forall y \cdot x \neq e \implies x = cons(y, z)$

$\forall y \cdot \forall z \cdot \text{First}(cons(y, z), y)$

Append

$\text{Append}(x, y, z)$ represents appending x to y and getting z

How do we write this definition in logic?

Case 1: x is the empty list.

$\forall x \cdot \forall y \cdot (x = e \implies \text{ Append}(x, y, y))$

$\forall y \cdot \text{ Append}(e, y, y)$

Case 2: x is not the empty list

$R_{first}(x, v_f) \implies$

$\quad R_{rest}(x, v_r) \implies$

$\quad\quad \text{Append}(v_r, y, v_a) \implies$

$\quad\quad\quad \text{Append}(x, y, cons(v_f, v_a))$

Using this simplification

$a \implies b \implies c \implies d \equiv (a \wedge b \wedge c) \implies d$

The above statement may be simplified.

$x \neq e \implies (R_{first}(x, v_f) \wedge R_{rest}(x, v_r) \wedge \text{Append}(v_r, y, v_a) \implies \text{Append}(x, y, cons(v_f, v_a))$

$\forall x \cdot \forall v_f \cdot \forall v_r \cdot \forall y \cdot \forall v_a \cdot (x \neq e \implies (R_{first}(x, v_f) \wedge R_{rest}(x, v_r) \wedge \text{Append}(v_r, y, v_a) \implies \text{Append}(x, y, cons(v_f, v_a)))$

Simplified using BL1 and BL2

$\forall v_r \cdot \forall y \cdot \forall v_a \cdot \forall v_f \cdot (\text{Append}(v_r, y, v_a) \implies \text{Append}(cons(v_f, v_r), y, cons(v_f, v_a)))$

$\forall x \cdot \forall y \cdot \forall z \cdot \forall w \cdot (\text{Append}(x, y, z) \implies \text{Append}(cons(w, x), y, cons(w, z)))$

Example: Append(x, y) is true iff x is a member of y

Case 1: Empty lists. $\forall x \cdot \neg$ Member(x, e)

Case 2: $\forall x \cdot \forall z \cdot (\text{Member}(x, cons(x, z)))$

$\forall x \cdot \forall w \cdot \forall z \cdot (\text{Member}(x, z) \iff \text{Member}(x, cons(w, z)))$

Example: Prove Member$(b, < a, b, c >)$

Member$(b, cons(a, cons(b, cons(c, e))))$

Member$(b, cons(b, cons(c, e)))$

True

Example: $\neg$Member$(b, < a >)$

Idea: Start with assuming Member$(b, < a >)$ and reach a contradiction. Therefore assumption is false and can conclude $\neg$Member$(b, < a >)$

Example: Reverse(x, y) is true iff y is the reverse of list x.

Case 1: Reverse(e, e) is true. $\forall x \cdot (x \neq e \implies \neg\text{Reverse}(x, e) \wedge \neg\text{Reverse}(e, x))$

Case 2: Reverse of a list is the reverse of the rest of the list appended onto the first element.

$\forall w \cdot \forall x \cdot \forall y \cdot \forall z \cdot (\text{Reverse}(x, y) \wedge \text{Append}(y, cons(w, e), z)) \implies \text{Reverse}(cons, w, x), z)))$

Reverse$(cons(b, e), cons(b, e))$

Given terms t_1, t_2, determine whether there are substitutions for the variables that make the terms the same. Finding such substitution is called **unification** of the terms.