# Predicate Logic

## Predicate Logic

Limitation of propositional logic – no way to talk about properties that apply to categories of objects, or relationships between those properties

Predicate logic – mathematical model for reasoning about predicates: functions that map variables to truth values

Predicate – boolean function whose value may be true or false depending on the arguments – generalization of propositional variables

## Quantifiers

- Additional operators to express truth values about predicates with variable arguments
  - Existential quantifier (there exists)
  - Universal quantifier (for all)

## Logical Expressions in Predicate Logic

- Similar to propositional logic expression with the following additions
  - An atomic formula is a logical expression
    - A predicate with all constant arguments is a ground atomic formula
    - A proposition is a predicate with no arguments and therefore is a ground atomic formula
    - A predicate with at least on variable argument is a nonground atomic formula
    - A literal is either an atomic formula or its negation
  - If L1 and L2 are logical expressions, then L1 AND L2, L1 OR L2, NOT L1, L1→L2, and L1≡L2 are logical expressions
  - If L1 is a logical expression, then (for all) L1 is a logical expression
  - If L1 is a logical expression, then (there exists) L1 is a logical expression
- Quantifiers have the highest precedence in logical expressions

## Bound and Free Variables

- Bound and free variables
  - Quantifiers introduce variables into logical expressions (variable x is bound to the closest enclosing quantifier)
  - Occurrence of variable not bound to a quantifier is free

## Evaluating Predicates

- Assign a real-word interpretation to P (e.g., addition, subtraction, equivalence) and a domain for P (i.e., possible values for arguments) – compute function P(x,y)
  - Can have an unbounded number of sets of arguments for which the predicate is true
- Consult a relational database containing pairs of values for x and y and the corresponding value of P(x,y)
  - Limited to finite domains

## Tautologies

- A tautology in predicate logic is a statement that is true regardless of the interpretation of predicates, and regardless of the bindings chosen for any globally unbound variables

## Evaluating Quantifiers

- Define
  - Domain over which the quantifier varies (the set of values for the predicate's arguments)
  - The interpretation (meaning) of the predicate
- If the domain of P is infinite, we don't have an algorithm that terminates to compute the value of P
- In many cases
  - the domain is finite
  - the equivalence of two expressions is required rather than whether an expression is true or not
- The truth of many statements with unbound variables is indeterminate
- The truth of many statements depends on the interpretation of predicates
- Some statements under some interpretations and proof systems are true but can't be proven

## Laws for Manipulating Quantifiers

- If a logical expression L is a tautology, all free variables in the tautology can be bound to universal quantifiers
- Moving NOT within a quantifier (analogous to DeMorgan's law)
  - NOT (forall x) L(x) ≡ (there exists x) (NOT L(x))
  - NOT (there exists x) L(x) ≡ (forall x) (NOT L(x))
- Moving quantifiers through AND and OR (where x is NOT a free variable in expression L1)
  - L1 AND (forall x) L2(x) ≡ (forall x) (L1 AND L2(x))
  - L1 AND (there exists x) L2(x) ≡ (there exists x)(L1 and L2(x)
  - L1 OR (forall x) L2(x) ≡ (forall x) (L1 OR L2(x))
  - L1 OR (there exists x) L2(x) ≡ (there exists x)(L1 OR L2(x))

## Prenex CNF Form

- Use laws for manipulating quantifiers to convert an expression to the form
  - (Q1 x1)(Q2 x2)…(Qn xn) L where all the quantifiers appear outside expression L
  - First, make sure all quantifiers refer to distinct variables (not found in other quantifiers or free variables)

## Proofs in Predicate Logic

- Similar to propositional logic
  - begin with a set of axioms (or hypotheses)
  - Use rules of inference to construct a sequence of expressions that follow from those axioms
- Inference rules – modus ponens, DeMorgan's law, substitution of equals, …
- Substitution rule – if a general fact (expression) is true, a specific instance is also true (variable substitution)
- Structure of a proof
  - Facts, or ground atomic formulae
  - Rules – conjunction of one or more atomic formulae that imply another atomic formula (general principles that can be applied to facts to prove new facts)

## Predicate Logic Proof Structure

- Assert a rule that is known to be true (i.e., the body of the rule implies the head of the rule
- Find facts that (via substitution) match the atomic formulae of the body of the rule
- Make consistent variable substitutions in the body and the head of the rule
- Assert the head (or goal) as proven

## Implication, Entailment, and Proof

- A→B – A implies B is a logical statement that may be true or false
- A|=B – A entails B is a meta-statement about truth. B is true in all models in which A is true. A→B is a tautology
- A|-B – B can be proven from A. This is a weaker meta-statement. It says that under some given set of proof rules and interpretations for predicates, if we are given A as premise, we can derive B

## Limits on Logic

- We have modeled computation as a "proof", and proceeded from facts (axioms) and rules to prove new facts (i.e., compute). Are there limits on what we can prove (compute)?
- The following are undecidable – no computer whatsoever can answer them
  - Goedel's Incompleteness Theorem [Kurt Goedel, 1931]
    - For number theory, there are expressions that are true but which cannot be proved to be true
  - The Halting Problem [Alan Turing, 1936]
    - There is no program that takes as input an arbitrary program and its input, and determines whether or not the program halts on that input

## Undecidability of the Halting Problem

- Assume halt(a,i) exists – returns TRUE if program a halts on input i, returns FALSE otherwise
- Create a new function trouble that returns TRUE if the input program does not halt on itself, and loops forever otherwise

```
function trouble(s)
    if (halt(s,s) == FALSE)
        return TRUE
    else
        loop forever
```

- If t represents the program trouble, does trouble(t) terminate?
  - Assume it does, then it doesn't (halt(t,t) returns FALSE)
  - Assume it doesn't, then it does (halt(t,t) returns TRUE)
  - Contradiction! in both cases. Therefore, the initial assumption that halt(a,i) exists must be incorrect

## Inherent Intractability

- P – class of problems solvable in polynomial time with no guessing
- NP – nondeterministic polynomial – if given a guess at a solution for some instance of size n, we can check that the guess is correct in polynomial time
- NP-Complete – problem in NP that can be proved to be as hard as any in NP
  - E.g., Satisfiability – is there a truth assignment that makes logical expression E true?
- NP-hard – problem not known to be in NP but as hard or harder than any problem in NP
  - E.g., the tautology problem – Is E a tautology?