FIRST-ORDER LOGIC

Adnan Shahzada

PROBLEMS WITH PROPOSITIONAL LOGIC

- Does not have any variables
- Cannot reason about a group of objects
- Cannot handle a domain with infinite objects

- We need
  - Predicates
  - Quantifiers

PREDICATE

- $P(x)$
  - P is the name of predicate
  - x is the variable and the argument of P
  - With n variables the predicate p has arity n

  - It has no truth values by itself
    - $P1(x) : 3+x = 8$
    - You can not associate any truth value with the above predicate without defining the domain of x

UNIVERSAL QUANTIFIER

- All men are mortal
  - $\forall x$ man(x) $\Rightarrow$ mortal(x)

- All cats are mammals
  - $\forall x$ cat(x) $\Rightarrow$ mammal(x)

- Universal Quantifier is good for writing general rules with universal quantifiers.

- $\forall x\ (x+3=8)$: This is not true by the rules of number theory if x is a member of the set of whole numbers
- $\forall x\ (x+1>x)$: This is true for all x where x is an integer

EXISTENTIAL QUNATIFIER

- Someone from Pakistan had won the Nobel prize
  - $\exists x$ pakistani(x) $\wedge$ wonNobelPrize(x)

- Some of the students in the AI class got an A.
  - $\exists x$ studentOfAI(x) $\wedge$ gotAnA(x)

- Existential quantifier is true if at least one element x from the specified domain satisfies the statement.

- $\exists x\ (x+3=5)$ is true
- $\exists x\ (x+1 < x)$ is false

SEMANTICS

- $\forall x\ p(x)$ is equivalent to p(a1) ^ p(a2) … p(an)
  - true for all objects in the domain of discourse D
- $\forall x\ p(x)$ is true iff p(x) is true for ALL x in D

- $\exists x\ p(x)$ is equivalent to p(a1) v p(a2) … p(an)
  - Only a particular p(x) has to be true if the entire propositional statement is true
- $\exists x\ p(x)$ is true iff p(x) is true for SOME x in D

Relation BETWEEN $\exists$ & $\forall$

- *Everyone dislikes Ahmad:*
  - $\forall x \neg Likes(x, Ahmad)$
  - $\neg \exists x\, Likes(x, Ahmad)$

- *Everyone likes Ahmad:*
  - $\forall x\, Likes(x, Ahmad)$
  - $\neg \exists x \neg Likes(x, Ahmad)$

- Rules

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

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

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

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

$\neg P \wedge \neg Q \equiv \neg (P \vee Q)$

$\neg (P \wedge Q) \equiv \neg P \vee \neg Q$

$(P \wedge Q) \equiv \neg(\neg P \vee \neg Q)$

$(P \vee Q) \equiv \neg(\neg P \wedge \neg Q)$

KR IN FIRST ORDER LOGIC

- You have to perform the following tasks for representing the knowledge by first order logic:
  - Identify the objects
  - Establish relationships among those identified objects
  - Encode the knowledge about these objects using relations in the form of rules
    - Grandfather(X,Y) <- father (X,Z) ^ father (Z,Y)

- Inference engine then can use that knowledge to infer new facts and knowledge.

KR IN FIRST ORDER LOGIC EXAMPLES

- Mother is a female parent:
  $\forall x \, \exists y \;\; female(y) \wedge parent(y,x) \Leftrightarrow mother(y,x)$
- Husband is male spouse
  $\forall w,h \; husband(h,w) \Leftrightarrow Male(h) \wedge spouse(h,w)$
- Disjoint categories e.g. male, female
  $\forall x \; male(x) \Leftrightarrow \neg \, Female(x)$
- Grandparent relation
  $\forall g,c \; grandparent(g,c) \Leftrightarrow \exists p \; Parent(g,p) \wedge Parent(p,c)$
- Sibling relation
  $\forall x,y \; Sibling(x,y) \Leftrightarrow x \neq y \wedge \exists p \; Parent(p,x) \wedge Parent(p,y)$

SKOLEMIZATION

- The process of removing existential quantifiers
- Example: someone likes Ahmad
  $\exists$x Likes (x, Ahmad)
- Get rid of existential quantifier by replacing x with a constant(called skolem constant) that doesn't exist in the KB
  $\exists$x Likes (Const1, Ahmad) {x/Const1}

- For complex examples you may have to use Skolem function. Consider this

$$\forall x[\exists y[mother\ (y, x)]]$$

If just replace the existential quantification with a constant we get

$$\forall x[mother\ (M302, x)]]$$

Which says M302 is everyone's mother.In this case we have to use a Skolem function rather than Skolem constant

$$\forall x[mother\ (f1(x), x)]]$$

Substitution

- Substitutions are also known as bindings
- A substitution is a list of variable-constant bindings of the form
  - θ = {var_1/val_1,v_2/g_2,…} where var_i are variables and val_i is the value substituted for var_i
  - If there is a clause C then Cθ is the atom we get when all occurrences of variables in θ are replaced by their ground terms
    - Example: C = parent(X,Y), θ= {X/Ali,Y/Akram},Cθ=father(Ali,Akram)

UNIFICATION

- It is the process or algorithm for determining substitutions so that we can make two predicates match
  - Example: unify(father(X,Akram),father(Ali,Akram))={X/Ali}

- FORMAL DEFINITION
  - Unify(C,D)= θ so that $C\theta = D\theta$
  - If no such θ exists then fail
- RULES: θ must satisfy the following:
  - No variable is bound to two different values θ={X/Ann,X/Bill} is invalid
  - No variable in θ is bound to a term that contains the variable itself θ = {X/f(X)} is invalid
- In unification you can do the following:
  - replace variable by constant
  - replace variable by variable
  - replace a variable by a function expression

UNIFICATION EXAMPLES

- Unify(f(a,X), f(a,b))
  = {X/b}
- Unify(f(a,X),f(Y,b))
  = {Y/a, X/b}
- Unify(f(a,b),f(a,b))
  = {}
- Unify(X,Y)
  = {X/Y}
- Unify(f(a,X),f(a,g(Y)))
  = {X/g(Y)}
- Unify(f(a,X),f(X,b))
  = fail (X is bound to a & b)
- Unify(f(a,b),f(a,c)) a,b,c are constants
  = fail (no match was found)
- Unify(f(a,b),g(a,X))
  = fail (different predicates)
- Unify(f(a,X),f(a,g(X)))
  = fail(X is bound to a term that contains itself)

Example

- hasWings(X) $\Rightarrow$ flies(X)
- hasWings(Tweety)

- θ = Unify (has_wings(X),has_wings(Tweety)) = {X/Tweety}

- So we can conclude that:
  - flies(Tweety)