

Predicate Logic

a.k.a. First Order Logic
(FOL)

Propositional Logic

- Propositional logic has limited expressive power.
 - All we can express are facts
 - Not really rules in the natural language sense.
 - We can't make a rule that expresses the notion that “a pit causes a breeze in all neighboring squares”
 - We have to add independent rules (facts) for each square.

First Order Logic

- Not just facts, much closer to natural language in expressive power.
- Assume the world contains
 - Objects (people, colors, numbers, ...)
 - Relations ($>$, is red, prime, part of, bigger than, owns, ...)
 - Functions (brother-of, one more than, ...)

Logics in general

- Ontology: the type of things that exist.
- Epistemology: the possible states, what an agent believes about each *fact*.

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	degree of truth $\in [0, 1]$	known interval value

Predicate Logic

- *Terms* represent specific objects in the *world* and can be constants, variables or functions.
- *Predicate Symbols* refer to a particular relation among objects.
- *Sentences* represent facts, and are made up of *terms*, *quantifiers* and *predicate symbols*.

Predicate Logic

- *Functions* allow us to refer to objects indirectly (via some relationship).
- *Quantifiers* and *variables* allow us to refer to a collection of objects without explicitly naming each object.

Some Examples

- Predicates: Brother, Sister, Mother, Father
- Objects: Joe, Sally, Roger, Jane
- Facts expressed as atomic sentences a.k.a. literals:

Father(Joe, Roger)

Mother(Sally, Roger)

Brother(Joe, Jane)

\neg Father(Joe, Roger)

Variables

- Variables can be used to represent unnamed objects.
 - In any sentence, all variables must be quantified
 - Either the variable can represent all objects, or the variable represents at least one object.

Variables and Universal Quantification

- Universal Quantification allows us to make a statement about a collection of objects:

$$\forall x \text{ Cat}(x) \Rightarrow \text{Mammal}(x)$$

All cats are mammals

“For All”

$$\forall x \text{ Father}(\text{Joe}, x) \Rightarrow \text{Mother}(\text{Sally}, x)$$

All of Joe’s kids are also Sally’s kids.

Variables and Existential Quantification

Existential Quantification allows us to state that an object does exist (without naming it):

“There exists” \rightarrow $\exists x \text{ Cat}(x) \wedge \text{Mean}(x)$
There is a mean cat.

$\exists x \text{ Father}(\text{Joe}, x) \wedge \text{Mother}(\text{Sally}, x)$

There is a kid whose father is Joe and whose mother is Sally

Nested Quantification

$\forall x, y \text{ Parent}(x, y) \Rightarrow \text{Child}(y, x)$

$\forall x \exists y \text{ Loves}(x, y)$

$\forall x [\text{Passtest}(x) (\exists x \text{ ComplainToDave}(x))]$

Functions

- Functions are terms - they refer to a specific object.
- We can use functions to symbolically refer to objects without naming them.
- Examples:

fatherof(x) *age(x)* *times(x,y)* *succ(x)*

Using functions

$\exists x \text{ Equal}(x, x)$

$\text{Equal}(\text{factorial}(0), 1)$

$\forall x \text{ Equal}(\text{factorial}(s(x)), \text{times}(s(x), \text{factorial}(x)))$

Representing facts with Predicate Logic - Example

- Marcus was a man
- Marcus was a Pompeian
- All Pompeians were Romans
- Caesar was a ruler.
- All Romans were either loyal to Caesar or hated him.
- Everyone is loyal to someone.
- Men only try to assassinate rulers they are not loyal to.
- Marcus tried to assassinate Caesar

Predicate Logic Knowledgebase

Man(Marcus)

Pompeian(Marcus)

$\forall x \text{ Pompeian}(x) \Rightarrow \text{Roman}(x)$

Ruler(Caesar)

$\forall x \text{ Romans}(x) \Rightarrow \text{Loyalto}(x, \text{Caesar}) \vee \text{Hate}(x, \text{Caesar})$

$\forall x \exists y \text{ Loyalto}(x, y)$

$\forall x \forall y \text{ Man}(x) \wedge \text{Ruler}(y) \wedge \text{Tryassassinate}(x, y) \Rightarrow$
 $\neg \text{Loyalto}(x, y)$

Tryassassinate(Marcus, Caesar)

Questions (Goals)

Was Marcus a Roman?

Was Marcus loyal to Caesar?

Who was Marcus loyal to?

Was Marcus a ruler?

Will the test be easy?

Quiz

Using the predicates:

Father(x,y) Mother(x,y) Brother(x,y) Sister(x,y)

Construct predicate logic facts
(sentences) that establish the following
relationships:

- GrandParent
- GrandFather
- GrandMother
- Uncle
- Cousin

Proof procedure for Predicate Logic: Resolution

- Same idea as with propositional logic, but a few added complexities:
 - conversion to CNF is much more complex.
 - Matching of literals requires providing a matching of variables, constants and/or functions.

$\neg \text{Skates}(x) \vee \text{LikesHockey}(x)$

$\neg \text{LikesHockey}(y)$

We can resolve these only if we assume x and y refer to the same object.

Predicate Logic and CNF

- Converting to CNF is harder - we need to worry about variables and quantifiers.
 1. Eliminate all implications \Rightarrow
 2. Reduce the scope of all \neg to single term.
 3. Make all variable names unique
 4. Move Quantifiers Left
 5. Eliminate Existential Quantifiers
 6. Eliminate Universal Quantifiers
 7. Convert to conjunction of disjuncts
 8. Create separate clause for each conjunct.

Eliminate Existential Quantifiers

- Any variable that is existentially quantified means we are saying there is some value for that variable that makes the expression true.
- To eliminate the quantifier, we can replace the variable with a function.
- We don't know what the function is, we just know it exists.

Skolem functions

$\exists y$ President(y)

We replace y with a new function $func$:

President($func()$)

$func$ is called a skolem function.

In general the function must have the same number of arguments as the number of universal quantifiers in the current scope.

Skolemization Example

$\forall x \exists y \text{ Father}(y,x)$

create a new function named foo and replace y with the function.

$\forall x \text{ Father}(\text{foo}(x),x)$

Predicate Logic Resolution

- We have to worry about the arguments to predicates, so it is harder to know when 2 literals match and can be used by resolution.
- For example, does the literal $\text{Father}(\text{Joe}, \text{Sally})$ match $\text{Father}(x, y)$?
- The answer depends on how we substitute values for variables.

Unification

- The process of finding a substitution for predicate parameters is called *unification*.
- We need to know:
 - that 2 literals *can* be matched.
 - the substitution that makes the literals identical.
- There is a simple algorithm called the *unification algorithm* that does this.

The Unification Algorithm

1. Initial predicate symbols must match.
2. For each pair of predicate arguments:
 - different constants cannot match.
 - a variable may be replaced by a constant.
 - a variable may be replaced by another variable.
 - a variable may be replaced by a function as long as the function does not contain an instance of the variable.

Unification Algorithm

- When attempting to match 2 literals, all substitutions must be made to the entire literal.
- There may be many substitutions that unify 2 literals, the *most general unifier* is always desired.

Unification Example

“substitute x for y”

- $P(x)$ and $P(y)$: substitution = (x/y)
- $P(x,x)$ and $P(y,z)$: $(z/y)(y/x)$ *y for x, then z for y*
- $P(x,f(y))$ and $P(\text{Joe},z)$: $(\text{Joe}/x, f(y)/z)$
- $P(f(x))$ and $P(x)$: can't do it!
- $P(x) \vee Q(\text{Jane})$ and $P(\text{Bill}) \vee Q(y)$:
 $(\text{Bill}/x, \text{Jane}/y)$

Unification & Resolution Examples

Father(Joe,Sally) \neg Father(Joe,x) \vee Mother(Jane,x)

Man(Marcus) \neg Man(x) \vee Mortal(x)

Loves(*father*(a),a) \neg Loves(x,y) \vee Loves(y,x)



This is a function

Predicate Logic Resolution Algorithm

- While no empty clause exists and there are clauses that can be resolved:
 - select 2 clauses that can be resolved.
 - resolve the clauses (after unification), apply the unification substitution to the result and store in the knowledge base.

Example:

$\neg \text{Smart}(x) \vee \neg \text{LikesHockey}(x) \vee \text{RPI}(x)$

$\neg \text{Canadian}(y) \vee \text{LikesHockey}(y)$

$\neg \text{Skates}(z) \vee \text{LikesHockey}(z)$

$\text{Smart}(\text{Joe})$

$\text{Skates}(\text{Joe})$

Goal is to find out if $\text{RPI}(\text{Joe})$ is true.

Man(Marcus)

Pompeian(Marcus)

\neg Pompeian(x_1) \vee Roman(x_1)

Ruler(Caesar)

\neg Romans(x_2) \vee Loyalto(x_2 , Caesar) \vee Hate(x_2 , Caesar)

Loyalto(x_3 , $f(x_3)$)

\neg Man(x_4) \vee \neg Ruler(y_1) \vee \neg Tryassassinate(x_4 , y_1) \vee
Loyalto(x_4 , y_1)

- PROVE: Tryassassinate(Marcus, Caesar)

Answering Questions

- We can also use the proof procedure to answer questions such as “who tried to assassinate Caesar” by proving:
 - $\text{Tryassassinate}(y, \text{Caesar})$.
 - Once the proof is complete we need to find out what substitution was made for y .

Computation

*s(x) is the integer
successor function*

Equal(y, y)

Equal(*factorial(s(x)), times(s(x), factorial(x))*)

...assume $s(_)$ and *times*($_, _$) can compute.

We can ask for 10!:

Equal(*factorial(10), z*)

Test Type Question

- The members of a bridge club are Joe, Sally, Bill and Ellen.
- Joe is married to Sally.
- Bill is Ellen's Brother.
- The spouse of every married person in the club is also in the club.
- The last meeting of the club was at Joe's house
- Was the last meeting at Sally's house?
- Is Ellen married?

Logic Programming - Prolog

- Prolog is a declarative programming language based on logic.
- A Prolog program is a list of facts.
- There are various predicates and functions supplied to support I/O, graphics, etc.
- Instead of CNF, prolog uses an implicative normal form:

$$A \wedge B \wedge \dots \wedge C \Rightarrow D$$

Prolog Example - Towers of Hanoi

```
hanoi(N) :- move(N,left,middle,right) .
```

```
move(1,A,_,C) :- inform(A,C) , ! .
```

```
move(N,A,B,C) :-  
    N1=N-1, move(N1,A,C,B) ,  
    inform(A,C) , move(N1,B,A,C) .
```

```
inform(Loc1,Loc2) :-  
    write("Move disk from" ,Loc1," to" , Loc2) .
```

```
hanoi(3)
```