Using Predicate Logic for Knowledge Representation

Dr. Krishnendu Guha

Assistant Professor (On Contract)

National Institute of Technology (NIT), Jamshedpur

Email: krishnendu.ca@nitjsr.ac.in

Representation of simple facts using logic

- Propositional logic is useful because it is simple to deal with and a decision procedure for it exists.
- Also, In order to draw conclusions, facts are represented in a more convenient way as,
- 1. Marcus is a man.
- • man(Marcus)
- 2. Plato is a man.
- • man(Plato)
- 3. All men are mortal.
- • mortal(men)

Problem of representation using Predicate Logic

- But propositional logic fails to capture the relationship between an individual being a man and that individual being a mortal.
- How can these sentences be represented so that we can infer the third sentence from the first two?
- Propositional logic commits only to the existence of facts that may or may not be the case in the world being represented.
- Moreover, It has a simple syntax and simple semantics. It suffices to illustrate the process of inference.
- Propositional logic quickly becomes impractical, even for very small worlds

PREDICATE LOGIC

- First-order Predicate logic (FOPL) models the world in terms of
- • **Objects**, which are things with individual identities
- • **Properties of objects** that distinguish them from other objects
- • **Relations** that hold among sets of objects
- • **Functions**, which are a subset of relations where there is only one “value” for any given “input”

- First-order Predicate logic (FOPL) provides
- • **Constants: a, b, dog33**. Name a specific object.
- • **Variables: X, Y**. Refer to an object without naming it.
- • **Functions:** Mapping from objects to objects.
- • **Terms:** Refer to objects
- • **Atomic Sentences:** in(dad-of(X), food6) **Can be true or false**, Correspond to propositional symbols P, Q.

Well Formed Formula

- A well-formed formula (*wff*) is a sentence containing no “free” variables.
- That is, all variables are “bound” by universal or existential quantifiers.
- (∀x)P(x, y) has x bound as a universally quantified variable, but y is free.

Quantifiers

- Universal quantification
- • $(\forall x)P(x)$ means that P holds for all values of x in the domain associated with that variable
- • E.g., $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)

- Existential quantification
- • $(\exists x)P(x)$ means that P holds for some value of x in the domain associated with that variable
- • E.g., $(\exists x)$ mammal(x) $\wedge$ lays-eggs(x)

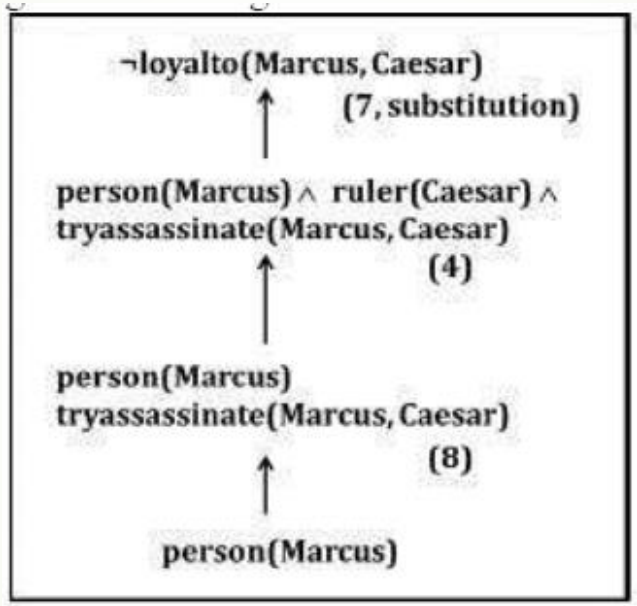
Example

- Shows the use of predicate logic as a way of representing knowledge.
- 1. Marcus was a man.
- 2. Marcus was a Pompeian.
- 3. All Pompeians were Romans.
- 4. Caesar was a ruler.
- 5. All Pompeians were either loyal to Caesar or hated him.
- 6. Everyone is loyal to someone.
- 7. People only try to assassinate rulers they are not loyal to.
- 8. Marcus tried to assassinate Caesar.

The facts described by well-formed formulas (*wffs*)
as follows:

1. Marcus was a man.
• man(Marcus)
2. Marcus was a Pompeian.
• Pompeian(Marcus)
3. All Pompeians were Romans.
• ∀x: Pompeian(x) → Roman(x)
4. Caesar was a ruler.
• ruler(Caesar)
5. All Pompeians were either loyal to Caesar or hated him.
• inclusive-or
• ∀x: Roman(x) → loyalto(x, Caesar) ∨ hate(x, Caesar)
• exclusive-or
• ∀x: Roman(x) → (loyalto(x, Caesar) ∧¬ hate(x, Caesar)) ∨ (¬loyalto(x, Caesar) ∧ hate(x, Caesar))
6. Everyone is loyal to someone.
• ∀x: ∃y: loyalto(x, y)
7. People only try to assassinate rulers they are not loyal to.
• ∀x: ∀y: person(x) ∧ ruler(y) ∧ tryassassinate(x, y) →¬loyalto(x, y)
8. Marcus tried to assassinate Caesar.
• tryassassinate(Marcus, Caesar)

- Now suppose if we want to use these statements to answer the question: ***Was Marcus loyal to Caesar?***
- Also, Now let's try to produce a formal proof, reasoning backward from the desired goal:
- ¬Ioyalto(Marcus, Caesar)



• The problem is that, although we know that Marcus was a man, we do not have any way to conclude from that Marcus was a person.
Also, We need to add the representation of another fact to our system, namely***:*** ∀ ***man(x) → person(x)***
• Now we can satisfy the last goal and produce a proof that Marcus was not loyal to Caesar.

Representing Instance and ISA Relationships

- Specific attributes **instance** and **isa** play an important role particularly in a useful form of reasoning called property inheritance.
- • The predicates instance and isa explicitly captured the relationships they used to express, namely class membership and class inclusion.

1. Man(Marcus).
2. Pompeian(Marcus).
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman(x).
4. ruler(Caesar).
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. $\forall x$: instance(x, Pompeian) $\rightarrow$ instance(x, Roman).
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. isa(Pompeian, Roman)
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).
6. $\forall x$: $\forall y$: $\forall z$: instance(x, y) $\wedge$ isa(y, z) $\rightarrow$ instance(x, z).

- The first part of the figure contains the representations we have already discussed. In these representations, class membership represented with unary predicates (such as Roman), each of which corresponds to a class.
- Asserting that P(x) is true is equivalent to asserting that x is an instance (or element) of P.
- The second part of the figure contains representations that use the ***instance*** predicate explicitly.
- The predicate ***instance*** is a binary one, whose first argument is an object and whose second argument is a class to which the object belongs.
- But these representations do not use an explicit ***isa*** predicate.
- Instead, subclass relationships, such as that between Pompeians and Romans, described as shown in sentence 3.
- The implication rule states that if an object is an instance of the subclass Pompeian then it is an instance of the superclass Roman.
- Note that this rule is equivalent to the standard set-theoretic definition of the subclasssuperclass relationship.
- The third part contains representations that use both the ***instance*** and ***isa*** predicates explicitly.
- The use of the ***isa*** predicate simplifies the representation of sentence 3, but it requires that one additional axiom (shown here as number 6) be provided.

Computable Functions and Predicates

- • To express simple facts, such as the following greater-than and less-than relationships:
  - gt(1,O) It(0,1) gt(2,1) It(1,2) gt(3,2) It(2,3)
- • It is often also useful to have computable functions as well as computable predicates.
  - Thus we might want to be able to evaluate the truth of gt(2 + 3,1)
- • To do so requires that we first compute the value of the plus function given the arguments 2 and 3, and then send the arguments 5 and 1 to gt.

- Consider the following set of facts, again involving Marcus:
- 1) Marcus was a man.
- man(Marcus)
- 2) Marcus was a Pompeian.
- Pompeian(Marcus)
- 3) Marcus was born in 40 A.D.
- born(Marcus, 40)
- 4) All men are mortal.
- x: man(x) → mortal(x)
- **5) All Pompeians died when the volcano erupted in 79 A.D.**
- **erupted(volcano, 79) ∧ ∀ x : [Pompeian(x) → died(x, 79)]**
- **6) No mortal lives longer than 150 years.**
- **x: t1: At2: *mortal(x) born(x, t1) gt(t2 – t1,150) → died(x, t2)***
- **7) It is now 1991.**
- ***now* = 1991**

- • So, Now suppose we want to answer the question “Is Marcus alive?”
- • The statements suggested here, there may be two ways of deducing an answer.
- • Either we can show that Marcus is dead because he was killed by the volcano or we can show that he must be dead because he would otherwise be more than 150 years old, which we know is not possible.
- • Also, As soon as we attempt to follow either of those paths rigorously, however, we discover, just as we did in the last example, that we need some additional knowledge. For example, our statements talk about dying, but they say nothing that relates to being alive, which is what the question is asking.
- So we add the following facts:
- 8) Alive means not dead.
- x: t: [alive(x, t) → ¬ dead(x, t)] [¬ dead(x, t) → alive(x, t)]
- 9) If someone dies, then he is dead at all later times.
- x: t1: At2*: died(x, t1) gt(t2, t1) → dead(x, t2)*
- So, based on these facts, we can answer the question “Is Marcus alive?” by proving: ¬ ***alive(Marcus, now)***


Thank
You
Merci
Obrigado
Asante
Grazie
Maake
Spasibo
Kiitos
Welalin
Matondo
Chokrane
Dankon
Mochchakkeram
Multumesc
Kia Ora
Ua Tsaug Rau Kqj
Nirringrazzjak
Dank Je
Spasibo
Vinaka
Arigato
Cam on ban
Terma Kasih
Juspaxar
Mamana
Salamat
Matour Nuwun
Ralbh Maith Agat