A Refresher Exercise, Part 1

Write predicate logic sentences for each of these facts:

1. Joe, Sally, Bill, and Ellen are all members of the Elm St. Bridge Club.

2. Sally is married to Joe.

3. Bill is Ellen's brother.

4. The spouse of every married person in the club is also in the club.

5. The last meeting of the club was at Joe's house.

One Possible Solution

Objects

- Joe Sally Bill Ellen ElmStBridgeClub

Predicates

- member(member, organization)
- married(spouse1, spouse2)
- brother(person, theBrother)
- lastMeetingAt(organization, location)

Functions

- houseOf(person)

Sentences

1. member(Joe, ElmStBridgeClub)
   member(Sally, ElmStBridgeClub)
   member(Bill, ElmStBridgeClub)
   member(Ellen, ElmStBridgeClub)

2. married(Sally, Joe)

3. brother(Ellen, Bill)

4. **forall** m, s
   married(s, m) **and**
   memberOf(m, ElmStBridgeClub)
   implies (s, ElmStBridgeClub)

5. lastMeetingAt(ElmStBridgeClub, houseOf(Joe))

A Refresher Exercise, Part 2

Given these facts:

1. member(Joe, ElmStBridgeClub)
   member(Sally, ElmStBridgeClub)
   member(Bill, ElmStBridgeClub)
   member(Ellen, ElmStBridgeClub)

2. married(Sally, Joe)

3. brother(Ellen, Bill)

4. **forall** m, s
   married(s, m) **and**
   memberOf(m, ElmStBridgeClub)
   implies (s, ElmStBridgeClub)

5. lastMeetingAt(ElmStBridgeClub, houseOf(Joe))

Derive the following statements:

1. The last meeting of the club was at Sally's house.

2. Ellen is not married.

Feel free to define any commonsense axioms that you need in order to do your derivations.

One Possible Solution

Commonsense Axioms

6. **forall** x, y
   married(x, y) **implies** married(y, x)

7. **forall** x, y
   married(x, y) **implies** houseOf(x) = houseOf(y)

8. **forall** x, y
   married(x, y) **implies not** brother(x, y)
   forall x, y
   married(x, y) **implies not** brother(y, x)

9. **forall** x, y, z
   married(x, y) **implies not** married(y, z)

Derivations

Where Are We?

We are exploring predicate logic with *modus ponens* as an inference rule.

- As a representation, logic gives a nice *formal language* in which to encode facts we believe to be true. Predicate logic allows us to represent a wide variety of facts about the world.

- As a reasoning mechanism, *modus ponens* lets us derive new facts in a straightforward, if cumbersome, way.

Using an inference rule to derive facts is a lot like using an operator to generate a successor to some state of the world.

In such a formulation, states are sets of sentences that describe a world, and an inference rule is an operator that allows you to generate successor states consisting of one more fact each.

Where Are We?

So, we can evaluate an inference rule in much the same way as we would a search strategy.

Modus ponens is:

- completeness Yes.
- optimality It depends.
- time It depends, but ....
- space It depends, but ....

Another interesting feature of logic is **soundness**. A sound inference rule derives only sentences that follow from the facts in the database. That is, if all sentences in a knowledge base are true, then it generates only true sentences.

Modus ponens is sound.

Problems with this Approach

One difficulty with automating this sort of reasoning comes from the two kinds of sentences in our knowledge base:

- Hoosier(Eugene)
- Hoosier(x) **implies** likes(x, Basketball)

Having two forms means that a program must be able to distinguish them and use them at the right times. This is an implementation problem.

Another difficulty with our current approach is that *modus ponens* only infers one fact from one fact and one implication. What if, for example, we wish to derive a new *implication*?

hoosier(x) **implies** likes(x, Basketball)
likes(x, Basketball) **implies** likes(x, March)
hoosier(x) **implies** likes(x, March)

This is a problem with the logic itself.

We need a new sort of inference rule.

Generalizing *Modus Ponens*

We can address both difficulties with one fix...

- **Represent facts as implications.**

  Hoosier(x)

  BECOMES

  True **implies** Hoosier(x)

  The semantics of **implies** make these sentences equivalent.

- **Modify *modus ponens* to work on implications.**

  p
  p implies q
  q

  BECOMES

  True **implies** p_i
  p_1 and ... p_i and ... p_n **implies** q
  p_1... p_{i-1} and p_{i+1}... p_n **implies** q

Generalized *Modus Ponens*

This rule allows us to derive an implication...

True	**implies** p_i
p_1 and ... p_i and ... p_n	**implies** q
p_1... p_{i-1} and p_{i+1}... p_n	**implies** q

allows:

a_1 and ... a_i and ... a_n	**implies** p_i	
p_1 and ... p_i and ... p_n	**implies** q	(p_i == a_i)
p_1... p_{i-1} and p_{i+1}... p_n	and	
a_1 and ... a_i and ... a_n	**implies** q	

A benefit of this approach is that the reasoner now can have a single goal. To derive q from (p_1 and ... p_i and ... p_n **implies** q), use generalized modus ponens to

drive p1 and p2 and ... p*n* **to** True

This is like recursion to a base case....