Using first order logic (Ch. 8-9)

Review: First order logic

In first order logic, we have objects and relations between objects

The relations are basically a list of all valid tuples that satisfy the relation

We can also have variables that represent objects often used in conjunction with quantifiers: $\forall, \exists$

First order logic

Let's translate English into first order logic:

"Everyone in class is sitting in a seat"
"If someone is sitting in a seat it is occupied"
"At least one seat is not occupied"
"No one is sharing a seat"

Objects: People (p1, p2, ...), Chairs (c1, c2, ...)
Relations: InClass(x), InSeat(x,y), Occupied(x)

person chair

First order logic

"Everyone in class is sitting in a seat"

$$\forall x \exists y \; InClass(x) \Rightarrow InSeat(x, y)$$

"If someone is sitting in a seat it is occupied"

$$\forall x \exists y \; InSeat(y, x) \Rightarrow Occupied(x)$$

"At least one seat is not occupied"

$$\exists x \; \neg Occupied(x)$$

"No one is sharing a seat"

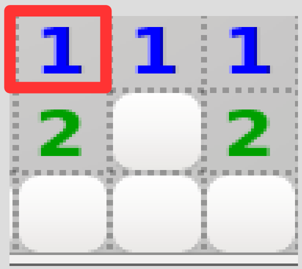
$$\forall x1, x2 \exists y1, y2 \; InSeat(x1, y1)$$

$$\wedge InSeat(x2, y2) \wedge x1 \neq x2 \Rightarrow y1 \neq y2$$

First order logic

To express the top left cell for mindsweep in propositional logic, we had to write:

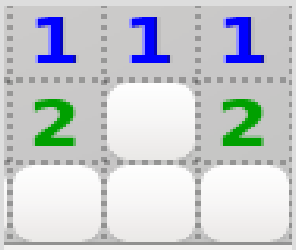
$$P1,1,1 \land \neg P1,1,2 \land \neg P1,1,3$$
$$\land \neg P1,1,4 \land \neg P1,1,5 \land \neg P1,1,6$$
$$\land \neg P1,1,7 \land \neg P1,1,8 \land \neg P1,1,B$$



How would you write the whole current knowledge for all 5 cells in first order logic? (not the game logic, just current state)

Hint: What are objects? Relations?

First order logic



First order logic:

$$One([1,1]) \wedge One([1,2]) \wedge One([1,3]) \wedge Two([2,1]) \wedge Two([2,3])$$

Then we just also need to say that cells can only have one number/bomb

$$\forall [x1,y1],[x2,y2],[x3,y3]...[x9,y9]\ One([x1,y1]) \wedge Two([x2,y2])$$
$$\wedge Three([x3,y3]) \wedge ... \wedge Eight([x8,y8]) \wedge Bomb([x9,y9])$$
$$\Rightarrow [x1,y1] \neq [x2,y2] \neq [x3,y3] \neq ... \neq [x9,y9]$$

Using First order logic

The rest of chapter 8 is boring, so we will skip (though good practice for logic representation)

We will go ahead into Ch. 9 and talk about how to use first order logic to query against

First we will look at how we can simplify some of the quantifiers

Universal instantiation

With a universal quantifier, $\forall$, this means you can replace it with any object

For example:
Objects = {Sue, Bob, Devin}
Sentence = $\forall x \ IsHuman(x)$

You can conclude:

$$\begin{aligned} &InHuman(Sue) \\ \wedge &InHuman(Bob) \\ \wedge &InHuman(Devin) \end{aligned}$$

Existential instantiation

With an existential quantifier, $\exists$, there is some object that makes this true...

So you give it a name of a new object (that is equal to an existing object)

Objects = {Spider, Dragon, Pangolin}
Sentence = $\exists x \ Mammal(x)$
You can conclude: $Mammal(M1)$
where $M1 = Spider \vee M1 = Dragon \vee M1 = Pangolin$

Convert to propositional logic

You can convert first order logic back into propositional logic by using instantiation

Objects = {Tree, Car}

Sentences: $\forall x \; Alive(x) \Rightarrow Reproduce(x)$

$Alive(Tree)$

Instantiation:

$Alive(Tree) \Rightarrow Reproduce(Tree)$

$Alive(Car) \Rightarrow Reproduce(Car)$

$Alive(Tree)$

Convert to propositional logic

Once you have this, you can treat each relation/object as a single proposition uniquely identified by the characters

$$Alive(Tree) \Rightarrow Reproduce(Tree)$$
$$Alive(Car) \Rightarrow Reproduce(Car)$$
$$Alive(Tree)$$

... could turn into:

$$AT \Rightarrow RT$$
$$AC \Rightarrow RC$$
$$AT$$

... and we could use our old techniques to ask information

Convert to propositional logic

This explanation glosses over two important facts... what?

Convert to propositional logic

This explanation glosses over two important facts... what?

1. Equals
2. Functions

(1) is easier to tackle as you can remove this when doing instantiation

You simply remove the invalid statements

Remove equality

Removing = after instantiation:

Object={A,B}

Sentence: $\forall x, y \; x \neq y \Rightarrow Different(x, y)$

Instantiation:

$$A \neq A \Rightarrow Different(A, A)$$
$$A \neq B \Rightarrow Different(A, B)$$
$$B \neq A \Rightarrow Different(B, A)$$
$$B \neq B \Rightarrow Different(B, B)$$

Remove conflicts:

$$True \Rightarrow Different(A, B)$$
$$True \Rightarrow Different(B, A)$$

Converting functions

I have skimmed on functions, but similar to math functions they can be applied repeatedly

Define: PlusPlus(x): $x \to x + 1$
PlusPlus(1) = 2
PlusPlus(PlusPlus(1)) = 3
... and so on (no limit to number of functions)

When converting to prop. logic, you have to apply functions everywhere possible...

Converting functions

This means the propositional logic conversion might have an infinite number of propositions

A theorem shows you only need a finite number of function calls to decide entailment

Step 1: See if entailed with no functions
Step 2: See if entailed with 1 function call
Step 3: See if entailed with 2 function calls
Step 4: ...

Converting functions

At some finite step, if entailment is possible it will be found

Unfortunately, how many is unknown so it is impossible to find if something is not entailed in the propositional logic (this is semi-decidable)

Even without functions if there are p k-ary relations with n objects, you get: O(p*n^k)

Unification

A <u>unification</u> is a substitution for variables that creates a valid sentence by specifying a map between variables and objects

For example, consider:

$$\text{Objects} = \{Sue,\ Alex,\ Devin\}$$

$$\forall x, y\ Sibling(x, y) \Rightarrow Sibling(y, x)$$

$$Sibling(Sue, Devin)$$

$$\neg Sibling(Devin, Alex)$$

What variables can we unify/substitute?

Unification

$$\text{Objects} = \{Sue,\ Alex,\ Devin\}$$
$$\forall x, y\ Sibling(x, y) \Rightarrow Sibling(y, x)$$
$$Sibling(Sue, Devin)$$
$$\neg Sibling(Devin, Alex)$$

First sentence is the only one with variables, there are 9 options (only 6 if x ≠ y)

One unification is {x/Sue, y/Devin}

We cannot say {x/Devin, y/Alex}, as this is creates a contradiction

Unification

$$\text{Objects} = \{Sue,\ Alex,\ Devin\}$$
$$\forall x, y\ Sibling(x, y) \Rightarrow Sibling(y, x)$$
$$Sibling(Sue, Devin)$$
$$\neg Sibling(Devin, Alex)$$

First sentence is the only one with variables, there are 9 options (only 6 if x ≠ y)

One unification is {x/Sue, y/Devin}
We cannot say {x/Devin, y/Alex}, as this is creates a contradiction

General modus ponens

We do not need to convert to propositional logic to use some rules of reasoning

Modus ponens can be applied even if there are variables, as long as we can unify them:

$$\forall x \; Large(x) \wedge Alive(x) \Rightarrow Dangerous(x)$$

$$\forall x \; Alive(x)$$

$$Large(Hippo)$$

We can unify the top sentence with {x/Hippo}, so we can conclude: $Dangerous(Hippo)$

General modus ponens

If you want to use this general modus ponens, finding the unification can be expensive

You basically need to try all substitutions, though you can store your data in smart ways to make look-up much more quickly

Using just general modus ponens, you can do basic inference with first order logic
(what is the problem??)

General modus ponens

Objects = {Cat, Dog, Frog, Rat, Sally, Jane}

$$\exists x \ Zodiac(x)$$
$$\forall x \ Alive(x) \Rightarrow Birthday(x)$$
$$\forall x \ Alive(x) \Rightarrow Eats(x)$$
$$\forall x, y \ Birthday(x) \Rightarrow Party(x, y)$$
$$\forall x \ Zodiac(x) \wedge Birthday(x) \Rightarrow Happy(x)$$
$$Alive(Sally)$$

Is Sally happy?

How about Party(Sally, Frog)?

General modus ponens

We can substitute {x/Sally} here with MP:

$\forall x \; Alive(x) \Rightarrow Birthday(x)$

To get: $Birthday(Sally)$

Then sub. {x/Sally, y/Frog} with MP here:

$\forall x, y \; Birthday(x) \Rightarrow Party(x, y)$

To get: $Party(Sally, Frog)$

However, we cannot tell if Sally is happy, as we cannot unify: $Zodiac(s1)$

$Birthday(Sally)$

General modus ponens

You try!

$$\forall x\ Meat(x) \land Make(Bread, x, Bread) \Rightarrow Sandwich(Bread)$$

$$\forall x, y\ OnGrill(x, y) \land Sandwich(y) \Rightarrow Grilled(y)$$

$$\forall x, y\ OnGrill(x, y) \land Meat(y) \Rightarrow Grilled(y)$$

$$\exists x\ Meat(x)$$

$$\forall x, y\ OnGrill(x, y)$$

$$\forall x, y, z\ Make(x, y, z)$$

$$Bread$$

Can you get Grilled(Bread)?

How about Grilled(Chicken)?

General modus ponens

You try!

$$\forall x\ Meat(x) \land Make(Bread, x, Bread) \Rightarrow Sandwich(Bread)$$

$$\forall x, y\ OnGrill(x, y) \land Sandwich(y) \Rightarrow Grilled(y)$$

$$\forall x, y\ OnGrill(x, y) \land Meat(y) \Rightarrow Grilled(y)$$

$$\exists x\ Meat(x)$$

$$\forall x, y\ OnGrill(x, y)$$

$$\forall x, y, z\ Make(x, y, z)$$

$$Bread$$

Can you get Grilled(Bread)? Yes

How about Grilled(Chicken)? No