Using first order logic (Ch. 8-9)

Modus ponendo ponens: \((p \rightarrow q) \land p \rightarrow q\)
If it rains, the road is wet, and it rains, so the road IS wet.

How to help the troopers to go asleep XD

Why do we have to listen such crap? Optimus promised us that we would read some good-night stories, not this sucking university book!
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: $\forall, \exists$
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)
First order logic

“Everyone in class is sitting in a seat”
$$\forall x \exists y \text{ InClass}(x) \Rightarrow \text{ InSeat}(x, y)$$

“If someone is sitting in a seat it is occupied”
$$\forall x \exists y \text{ InSeat}(y, x) \Rightarrow \text{ Occupied}(x)$$

“At least one seat is not occupied”
$$\exists x \neg \text{ Occupied}(x)$$

“No one is sharing a seat”
$$\forall x_1, x_2 \exists y_1, y_2 \text{ InSeat}(x_1, y_1) \wedge \text{ InSeat}(x_2, y_2) \wedge x_1 \neq x_2 \Rightarrow y_1 \neq y_2$$
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:

\[ \text{One}([1, 1]) \land \text{One}([1, 2]) \land \text{One}([1, 3]) \land \text{Two}([2, 1]) \land \text{Two}([2, 3]) \]

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

\[ \forall [x_1, y_1], [x_2, y_2], [x_3, y_3] \ldots [x_9, y_9] \text{ One}([x_1, y_1]) \land \text{Two}([x_2, y_2]) \land \text{Three}([x_3, y_3]) \land \ldots \land \text{Eight}([x_8, y_8]) \land \text{Bomb}([x_9, y_9]) \]

\[ \Rightarrow [x_1, y_1] \neq [x_2, y_2] \neq [x_3, y_3] \neq \ldots \neq [x_9, y_9] \]
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: $InHuman(Sue)$
$\land InHuman(Bob)$
$\land InHuman(Devin)$
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 $M1 = Spider \lor M1 = Dragon \lor M1 = Pangolin$
You can convert first order logic back into propositional logic by using instantiation.

Objects = \{\text{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

\[
\begin{align*}
\text{Alive}(\text{Tree}) & \Rightarrow \text{Reproduce}(\text{Tree}) \\
\text{Alive}(\text{Car}) & \Rightarrow \text{Reproduce}(\text{Car}) \\
\text{Alive}(\text{Tree}) &
\end{align*}
\]

... could turn into:

\[
\begin{align*}
\text{AT} & \Rightarrow \text{RT} \\
\text{AC} & \Rightarrow \text{RC} \\
\text{AT} &
\end{align*}
\]

... 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: \( \text{PlusPlus}(x) : x \rightarrow x + 1 \)
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: $O(p*n^k)$
A unification is a substitution for variables that creates a valid sentence by specifying a map between variables and objects.

For example, consider:

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

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 \neq y)

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

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 \(x \neq y\))

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:

\[
\forall x \text{ Large}(x) \land \text{ Alive}(x) \implies \text{ Dangerous}(x) \\
\forall x \text{ Alive}(x) \\
\text{Large(Hippo)}
\]

We can unify the top sentence with \{x/Hippo\}, so we can conclude: \text{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??)
Objects = \{\text{Cat, Dog, Frog, Rat, Sally, Jane}\}

\exists x \ \text{Zodiac}(x)
\forall x \ \text{Alive}(x) \implies \text{Birthday}(x)
\forall x \ \text{Alive}(x) \implies \text{Eats}(x)
\forall x, y \ \text{Birthday}(x) \implies \text{Party}(x, y)
\forall x \ \text{Zodiac}(x) \land \text{Birthday}(x) \implies \text{Happy}(x)
\text{Alive}(\text{Sally})

Is Sally happy?
How about \text{Party}(\text{Sally, Frog})?
General modus ponens

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

\[ \forall x \; \text{Alive}(x) \Rightarrow \text{Birthday}(x) \]

To get: \( \text{Birthday}(Sally) \)

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

\[ \forall x, y \; \text{Birthday}(x) \Rightarrow \text{Party}(x, y) \]

To get: \( \text{Party}(Sally, Frog) \)

However, we cannot tell if Sally is happy, as we cannot unify:

\[ \text{Zodiac}(s1) \quad \text{Birthday}(Sally) \]
General modus ponens

You try!

\[ \forall x \, \text{Meat}(x) \land \text{Make}(\text{Bread}, x, \text{Bread}) \Rightarrow \text{Sandwich}(\text{Bread}) \]

\[ \forall x, y \, \text{OnGrill}(x, y) \land \text{Sandwich}(y) \Rightarrow \text{Grilled}(y) \]

\[ \forall x, y \, \text{OnGrill}(x, y) \land \text{Meat}(y) \Rightarrow \text{Grilled}(y) \]

\[ \exists x \, \text{Meat}(x) \]

\[ \forall x, y \, \text{OnGrill}(x, y) \]

\[ \forall x, y, z \, \text{Make}(x, y, z) \]

\[ \text{Bread} \]

Can you get Grilled(Bread)?

How about Grilled(Chicken)?
You try!

\( \forall x \, \text{Meat}(x) \land \text{Make}(Bread, x, Bread) \Rightarrow \text{Sandwich}(Bread) \)

\( \forall x, y \, \text{OnGrill}(x, y) \land \text{Sandwich}(y) \Rightarrow \text{Grilled}(y) \)

\( \forall x, y \, \text{OnGrill}(x, y) \land \text{Meat}(y) \Rightarrow \text{Grilled}(y) \)

\( \exists x \, \text{Meat}(x) \)

\( \forall x, y \, \text{OnGrill}(x, y) \)

\( \forall x, y, z \, \text{Make}(x, y, z) \)

\text{Bread}

Can you get Grilled(Bread)? Yes
How about Grilled(Chicken)? No