Week 8

Notes on first order logic structures

Instructor: Matteo de Ceglie

07 May 2020

Last week we introduced first order languages, and begin our exploration of first order logic, a much more powerful logic than propositional logic. However, we didn't touched on the problem of deciding when a certain formula in first order logic is true or false. We know that in the case of propositional logic, the method of the truth tables was enough. Given any formula, even very complex one like the following:

$$((\varphi \to (\psi \vee (\chi \wedge \rho))) \leftrightarrow (\psi \wedge \neg(\vartheta \to \neg\varphi))) \vee (\chi \to (\neg\rho \wedge \tau))$$

(and you can easily imagine more complex ones!) you know that you could determine the truth value of the whole formula. Sure, the method of truth tables can be tedious at times (in this case, with 5 different variables, you need to check 32 cases!), but it always arrives a single and precise result.

How about formulas in first order logic? Take for example the following:

$$\forall x[P(x)]$$

This formula is, admittedly, very simple. Can we use truth tables to see if it is a tautology, contingent or a contradiction? Sadly, we cannot. The reason is that there cannot be a truth tables that accounts for all the possible meaning that x takes in this case. Suppose that P means "being a parent". Our formula would then mean "Every x is a parent". How do we check that? And what happens when the formula is just a little bit more complex like $\forall x[P(x) \to Q(x)]$? We surely cannot check every possibility, and build an infinite truth table! Instead, we are going to *build a model* (called also a structure) for that formula (we have already seen a glimpse of what this means last week, when I introduced in a very informal way the concept of substitution and domain for the translations).

Notice that this is needed in every case. For example, suppose we need to evaluate the following formula:

$$\forall x[\forall y[P(x) \wedge \neg P(y)]$$

At first glance this would seem a contradiction, since inside the parenthesis we have a formula of the form $\varphi \wedge \neg\varphi$. However, notice that in one case we have $P(x)$, and in the other $P(y)$. Since the variables are actually different, we cannot use our knowledge of contradictions and tautologies from propositional logic.

What are the steps to build a model? First of all notice that the language of first order logic is *uninterpreted*, that is, there is no meaning attached to its symbols, variables and constants. The formula

$$\forall x[\exists y[P(x) \to Q(x, y)]]$$

has no attached meaning, and can be translated as "For all x, there exists and y such that if $P(x)$, then $Q(x, y)$. The only way to understand this formula is to provide a translation key, and explicitly say what the constants and relations symbols in the formula mean. Moreover, we also need to say on which things we are quantifying, i.e. what things can the variables $x, y, \ldots$ be (for example all the humans, all the balls in a ball pit, the natural numbers etc.). In our example above, we can say that the relation $P(x)$ means "being a ball", the relation $Q(x, y)$ means "x is

of the same brand as y" and the domain on which x and y varies are all the balls in the ball pit outside my house. Only at this point we can check if that formulas is true: we go outside, take all the red balls in the pit, and for every red ball we search for another ball of the same brand (not necessaries red, but could also be red).

We can make this a little bit more precise, and give the following definition:

Definition 1 (First order Model)**.** A *model* $\mathcal{M}$ for the language of first order logic consists of the following elements:

- A non-empty set $\mathcal{D}$, called the *domain*;
- For every constant symbol of the language, an interpretation of that symbol in the domain (we wrote $c^{\mathcal{M}}$ to say the constant c is being interpreted in the model $\mathcal{M}$);
- For every relation symbol of the language, and interpretation of that symbol in the domain (we wrote $R^{\mathcal{M}}$ to say the constant c is being interpreted in the model $\mathcal{M}$);

This definition is quite straightforward, and it is just a rewriting of what I introduced above. However, there is a little detail worth noting: we do not admit empty domain. Why is that? The reason behind this stipulation is that we need to ensures that our laws of propositional logic are still true in any model. An empty domain would make any sentence trivially true for that model. On the other hand, in a non empty domain the following sentence (or any other law of propositional logic) is still true:

$$\exists x[P(x) \vee \neg P(x)].$$

So when asked to give a model for a formula we need to provide all those things. For example, for our formula $\forall x[\exists y[P(x) \to Q(x,y)]]$ we can build the following model $\mathcal{M}$:

- $\mathcal{D}$ = all the fish in the ocean;
- $P^{\mathcal{M}}(x)$ = "x is a salmon";
- $Q^{\mathcal{M}}(x,y)$ = "y is bigger than x"

Notice that, just like in the case of the translations, the same formula can be interpreted in two completely different models.

The method showed above to build models is the basic one. In case of more complex formulas, we need to consider when a formula is *satisfied*. The easiest definition is that a formula φ is satisfied if and only if it has a model (i.e. if and only if we can find a domain, an interpretation of the constants and of the relations such that φ is true). If $\mathcal{M}$ is such a model, we write $\mathcal{M} \models \varphi$. If a model $\mathcal{M}$ does not satisfy the formula φ, then we write $\mathcal{M} \not\models \varphi$. This relation of satisfaction is probably one of the most important relations that one can find in logic. The main feature of this relation is that it is compositional: the satisfaction of a formula φ is based on the satisfaction of it's sub-formulas. We can thus define this relation:

Definition 2 (Satisfaction)**.** Let φ and ψ be formulas in first order logic, and be $\mathcal{M}$ a first order model. Then $\mathcal{M} \models \varphi$ if and only if:

1. If φ is a contradiction, then for every model $\mathcal{M}$, $\mathcal{M} \not\models \varphi$.
2. $\mathcal{M} \models \neg\varphi$ if and only if $\mathcal{M} \not\models \varphi$;
3. $\mathcal{M} \models \varphi \wedge \psi$ if and only if $\mathcal{M} \models \varphi$ and $\mathcal{M} \models \psi$;
4. $\mathcal{M} \models \varphi \vee \psi$ if and only if $\mathcal{M} \models \varphi$ or $\mathcal{M} \models \psi$ (or both);
5. $\mathcal{M} \models \varphi \to \psi$ if and only if $\mathcal{M} \not\models \varphi$ or $\mathcal{M} \models \psi$ (or both);
6. $\mathcal{M} \models \forall x[\varphi]$ if and only if, for every possible interpretation of x in the model, $\mathcal{M} \models \varphi$;
7. $\mathcal{M} \models \exists x[\varphi]$ if and only if there is an interpretation of x in the model such that $\mathcal{M} \models \varphi$.

As you can see, this definition follows the already known pattern of the truth tables for the connectives. Moreover, it adds a semantics meaning to the quantifiers. Following this definition it is possible to show that the following laws (called De Morgan's Law for the quantifiers) are true:

1. $\forall x[P(x)] \leftrightarrow \neg\exists x[\neg P(x)]$;
2. $\exists x[P(x)] \leftrightarrow \neg\forall x[\neg P(x)]$.

They both instantiate the counterexample method: to be true that all x are P means not being able to find an x that is not P; to be true that there exists an x that is P means that it is false that all x are not P.

We can now put all of this together and ask ourselves if there exists a model for the following formula:

$$\exists x[A(z,c)] \rightarrow \forall y[A(y,x) \vee A(z,x)].$$

We can try to do that and build a model from scratch, checking all the possibilities. However that way would we long, so we are better off using the above definition. What we need to do is trying to prove that

$$\mathcal{M} \models \exists x[A(z,c)] \rightarrow \forall y[A(y,x) \vee A(z,x)]$$

From the previous definition, this means proving one of the following:

1. $\mathcal{M} \models \forall y[A(y,x) \vee A(z,x)]$
2. $\mathcal{M} \not\models \exists x[A(z,c)]$.

Proving both of them, while possible, would be an overkill, and we do not need to do that. In this spirit, we can further divide our first formula, since we need to prove just one of the following:

1. $\mathcal{M} \models \forall y[A(y,x)]$;
2. $\mathcal{M} \models \forall y[A(z,x)]$.

So we will work on only one of the following formulas:

1. $\mathcal{M} \models \forall y[A(y,x)]$;
2. $\mathcal{M} \models \forall y[A(z,x)]$.
3. $\mathcal{M} \not\models \exists x[A(z,c)]$.

The best strategy in this cases is to try and build the model for the existential quantifiers, since it is easier and quicker (they need only one case to be checked, instead of all the possible cases in the domain). Sadly, in this example all the formulas involves universal quantifier (remember that the negation of an existential quantifier is equivalent to the universal quantifier with the negation inside its scope), so the choice does not really matter. We will go forward with the third formula:

$$\mathcal{M} \not\models \exists x[A(z,c)].$$

This means we need to build a model that *does not satisfy* $\exists x[A(z,c)]$. Why choose this one? In general, it is better to choose a formula with constants in it, since it means less cases to check. A very simple one would suffice, since the simpler the model, the easiest checking it! Our model $\mathcal{M}$ can look like this:

- $\mathcal{D} = \{1,2,3\}$;
- $c^{\mathcal{M}} = 3$;
- $A^{\mathcal{M}}(x,y) = x > y$.

The domain of this model contains only two elements: the numbers $1, 2$ and 3 (since in the whole formulas there are 3 different variables, it is better to use at least 3 elements in the domain, to avoid complications). The variables of our formula can thus vary on this domain, while to the constant c we assign the value 3 (it is a constant, so it is going to be always 3). Finally, to the relation $A(x, y)$ we assign the meaning "x is strictly bigger than y" (this means that they cannot be equal). Does this model satisfy our formula? Obviously not, since since in all cases our variable z is not strictly bigger than our constant:

- when $z = 1$, we have $1 > 3$, and this is false;
- when $z = 2$, we have $2 > 3$, and this is false;
- when $z = 3$, we have $3 > 3$, and this is also false.

Thus, for the definition of satisfaction, we know that the model $\mathcal{M}$ satisfies the whole formula $\exists x[A(z, c)] \to \forall y[A(y, x) \vee A(z, x)]$, since it does not satisfy its sub-formula $\exists x[A(z, c)]$. (You will not that in this case I chose a model with number and an arithmetic relation like $>$. Do not be intimidated by this, this is actually not real mathematics and you do not need any knowledge outside the very basic arithmetic done in elementary school. However, when I ask you to build a model, ***it is not necessary to do use numbers like I did***, and you can use fish, Socrates, fish that sits near Socrates but not Plato and whatever you prefer.)