Peter Dillinger

## Intro to First Order Logic

Today we begin talking about logics more sophisticated and powerful than Boolean logic. I will introduce the standard "first order logic" or "predicate logic" and relate that to the ACL2 logic.

Consider a small mathematical statement I might make:

$$2x \geq x$$

Is this true? You might say, "yes, it is true," but it depends on what x can be. If x can be a negative number, this statement is not true. In this sense, mathematicians are rather sloppy: there are often unwritten assumptions in the statements they make. They can say, "You know what I mean," and they are often right.

Computers, however, only understand what we tell them, so we need to be more precise. First order logic has ways of making this more precise. First of all, we should write

$$\forall x \;\; 2x \geq x$$

This reads as, "For all x, two times x is greater than or equal to x." More precisely, this means, "For any value assigned to the variable x, two times x is greater than or equal to x." This ∀x is known as **universal quantification**.

But this is not quite what we want because for the whole statement to be true, the part inside the "for all" has to be true for *any* value of x, including negative numbers. So in fact,

$$x = -1$$

is a **counterexample** to the above, because it demonstrates how it can be false.

To make a true statement, we probably want to restrict this to natural numbers. Well, there is only one kind of universal quantification, but we have studied something in boolean logic has that allows us to state that something must be true when something else is true. What we want to say here is, "for all x, if x is a natural number, then 2x ≥ x." What we need is boolean implication:

$$\forall x \;\; x \in \mathbb{N} \rightarrow 2x \geq x$$

$x \in \mathbb{N}$ is the way mathematicians say that x is a natural number (more precisely, in the set of natural numbers). The ACL2 equivalent would be `(natp x)`.

Recall that if the hypothesis of an implication (here $x \in \mathbb{N}$) is false, the formula is true. Thus, the statement inside the "for all" is true for all x values that are not natural numbers. For all natural numbers x, the body of the "for all" is true exactly when 2x ≥ x is true. (Recall that *false* → p is equivalent to *true* and *true* → p is equivalent to p.)

Well, under standard arithmetic, 2x ≥ x is true for all natural numbers. That makes this statement a **theorem**. It is not a **tautology**, though. Here is the distinction: a theorem is a statement that is always true in a particular logic *and* a particular theory; a tautology is always true in a particular logic, regardless of the theory.

That does not help much without understanding the difference between a **logic** and a

**theory**. Well, the distinction does not have much practical significance, and many times it is not clearly defined. For first order logic, the distinction has much to do with the distinction between formulas and expressions.

A **formula** is something that is either true or false, possibly depending on the meaning of relations/predicates and expressions in it. Examples of formulas in first order logic are

| | |
|---|---|
| $x = y$ | (Relations such as equality) |
| $\forall x \ x = y$ | (Quantified formulas) |
| $x < y \ \wedge \ \neg(y < z)$ | (Formulas combined with boolean logic operations) |

An **expression** is something that corresponds to a value. Which value can depend on assignments to variables and the meaning of constants, functions, and relations/predicates. Examples in first order logic are

| | |
|---|---|
| x | (Variable reference) |
| 42 | (Constant expression) |
| f(x) | (Function application) |
| x + 42 | (Also a function application; logically, addition is a function) |

Now that we have that, a **theory** constrains the meaning of constants, functions, and relations/predicates. For example, there are standard theories of arithmetic that give us functions like + and log, relations like <, and constants like 16. None of these are "built in" to first order logic.

The **logic** gives us the meaning of quantifiers and boolean logic operations, and usually one relation is built in to a logic: equality (=).

Let's get back to how this relates to **theorems** and **tautologies**.

$3 = 3 \ \wedge \ 4 = 4$

is a tautology because equality and conjuction (/\) are built into first order logic, so no matter what a particular theory says about the constants 3 and 4, this formula is true. Similarly,

$\forall y \ y = y$

is a tautology, because the basic meaning of quantification, equality, and variable reference are built into the logic. The theory can constrain what values constitute "all values," but in first order logic, they are all equal to themselves.

However,

$3 < 4$

is not a tautology. It is a theorem in standard theories of arithmetic, but one could construct a theories in which this is always false, or even unknown. Thus, it is not a tautology.

What about x = x? Well, there is one more requirement we have for something to be a theorem. It cannot have any free variables. Any variables used must be quantified "above" those uses, just like variables in ACL2 must be bound by a LET, LET*, or function parameter list. Let us introduce this along with some other terms...

A logical formula that has no free variables is a **proposition**.

A proposition that one suspects is true is a **conjecture**.

A proposition that is always true in a particular logical theory is a **theorem**.

In first order logic, we have one other kind of quantifier, the **existential quantifier**, ∃. For example,

$$\exists x \; x \in \mathbb{Z} \; \wedge \; x < 0$$

$x \in \mathbb{Z}$ means x is an integer, so the whole thing would read as, "There exists an x for which x is in the set of integers and x is less than zero." It is true if we can give x some value which is both an integer and less than zero. We can. x=-1 is a **witness** that shows the existential is true in standard arithmetic. Thus, the proposition is a theorem.

There is a close relationship between the two types of quantifiers. We can use some identities to convert between them:

$$\neg\forall x \; \varphi \quad \text{is equivalent to} \quad \exists x \; \neg\varphi$$

$$\neg\exists x \; \varphi \quad \text{is equivalent to} \quad \forall x \; \neg\varphi$$

Here I am using the greek letter φ (phi) to stand for any formula. To help us understand the first identity above, let φ be "if x is a CS U290 student, then they were in class on October 9". We might write that as

$$\text{csu290\_student?}(x) \rightarrow \text{in\_class\_oct9?}(x)$$

where csu290_student? and in_class_oct9? are predicates. The first identity above claims that

$$\neg\forall x \; \text{csu290\_student?}(x) \rightarrow \text{in\_class\_oct9?}(x)$$

which states that, "it is not true that every CS U290 student was in class on October 9," is the same as

$$\exists x \; \neg(\text{csu290\_student?}(x) \rightarrow \text{in\_class\_oct9?}(x))$$

From boolean logic, we know that $\neg(p \rightarrow q)$ is the same as $p \wedge \neg q$. Thus, this existential formula is the same as

$$\exists x \; \text{csu290\_student?}(x) \; \wedge \; \neg \; \text{in\_class\_oct9?}(x)$$

which states that, "there exists a CS U290 student who was not in class on October 9." Conceptually, this is the same thing! Not everyone was in class. (Were you?)

Consider this example: (from second identity, slightly different φ)

$$\neg\exists x \; \text{csu290\_student?}(x) \; \wedge \; \text{in\_class\_oct9?}(x)$$

That would mean there does not exist a CS U290 student who was in class on October 9. But that's the same as saying everyone who is a CS U290 student was *not* in class on October 9:

$$\forall x \; \neg(\text{csu290\_student?}(x) \; \wedge \; \text{in\_class\_oct9?}(x))$$

$$\forall x \; \neg \; \text{csu290\_student?}(x) \; \vee \; \neg \; \text{in\_class\_oct9?}(x)$$

$$\forall x \; \text{csu290\_student?}(x) \rightarrow \neg \; \text{in\_class\_oct9?}(x)$$

You have probably noticed that it is pretty common for a universal quantifier to have an implication (→) inside of it and for an existential quantifier to have a conjuction (∧) inside of it. This isn't always the case, but they are very useful in restricting the kinds of values you are really interested in for the quantification.

## A Simple Conjecture

Let us try to write a logical formula for something we know from mathematics:

> "There is no largest integer."

How do I write that as a formula? Well, let's rephrase it slightly:

> "There does not exist an integer that is the largest."

We know how to write "There does not exist ...":

$$\neg\exists x \ \ldots$$

Now we need a formula that says x is an integer and x is largest. Well, we've got half of it:

$$\neg\exists x \ x \in \mathbb{Z} \ \bigwedge \ \ldots$$

For x to be largest, it has to be greater than or equal to every other integer. How do I write a formula that must be true for lots of cases—even an infinite number of cases? (Universal quantification)

$$\neg\exists x \ x \in \mathbb{Z} \ \bigwedge \ (\forall y \ y \in \mathbb{Z} \ \rightarrow \ x \geq y)$$

That says that, "there does not exist an x that is both an integer and, for all y that are integers, x is greater than or equal to y."

We can use the identities on quantifiers along with our boolean logic identities to change this into equivalent forms:

$$\forall x \ \neg(x \in \mathbb{Z} \ \bigwedge \ (\forall y \ y \in \mathbb{Z} \ \rightarrow \ x \geq y))$$

$$\forall x \ \neg \ x \in \mathbb{Z} \ \bigvee \ \neg(\forall y \ y \in \mathbb{Z} \ \rightarrow \ x \geq y)$$

$$\forall x \ x \in \mathbb{Z} \ \rightarrow \ \neg(\forall y \ y \in \mathbb{Z} \ \rightarrow \ x \geq y)$$

$$\forall x \ x \in \mathbb{Z} \ \rightarrow \ \exists y \ \neg(y \in \mathbb{Z} \ \rightarrow \ x \geq y)$$

$$\forall x \ x \in \mathbb{Z} \ \rightarrow \ \exists y \ y \in \mathbb{Z} \ \bigwedge \ \neg \ (x \geq y)$$

Now, assuming we are in a standard theory of arithmetic, we can use the fact that ¬(x ≥ y) is equivalent to y > x:

$$\forall x \ x \in \mathbb{Z} \ \rightarrow \ \exists y \ y \in \mathbb{Z} \ \bigwedge \ y > x$$

This is formula is rather simple to read and if we think about its meaning, it seems it should be equivalent to our original statement of the proposition: "For every integer, there exists an integer that is larger."

We aren't yet going into the details of proof, but how would we prove this in mathematics? Well, to prove an existential, we need a witness. In this case, the existential (on y) is inside a universal quantifier (on x). Thus, we need a witness for each x for which the existential must be true.

Well, the formula in the universal quantifier is an implication, $x \in \mathbb{Z} \rightarrow \exists y \ldots$. Thus, if x is not an integer, then the body is true regardless of the existential formula. But if x is an integer, then the existential formula must be true for the body to be true. Thus, we need to find witnesses to the existential for each possible integer value of x.

If x is an integer, can you name an integer greater than it?

How about x + 1? (Yes, it works.) In fact, I can eliminate the existential quantifier by plugging in x + 1 for each y under it in the formula:

$$\forall x \; x \in \mathbb{Z} \rightarrow (x+1 \in \mathbb{Z} \wedge x+1 > x)$$

Our knowledge of mathematics tells us that this is a theorem in standard theories of arithmetic. (An integer plus one is also an integer and greater than the original integer.)

## The ACL2 Logic

ACL2's logic is "quantifier-free," which means quantifiers don't appear in ACL2 logical formulas. There is quantification, though, because free variables are understood to be universally quantified. Universal quantification is implicit. For example,

```
(or (atom x) (consp x))
```

means that for all values of x, x satisfies `atom` or x satisfies `consp`. In logic and theory of ACL2, this is a theorem, because there are no values for x that make this false (nil).

What about existential quantification? In the example of there being no largest integer, we saw that we could get rid of the existential quantifiers in a formula if we can come up with the right witnesses. We can write the final formula in ACL2:

```
(implies (integerp x)
         (and (integerp (+ x 1))
              (> (+ x 1) x)))
```

Wouldn't it be more convenient if we could just write the version with existential quantification? What is the cost of doing so?

The answer has to do with one of the central design ideas for ACL2. When we are forced to use witnesses as opposed to existential quantifiers, that helps immensely in proving theorems automatically. When we just say "there is some value somewhere for which this formula is true; from among infinite possibilities, go figure out what it is," that is much harder to deal with than "here is the value for which this formula is true; verify that I'm telling the truth."

There is another problem with writing quantifiers: they are not programming language constructs; they are not computable. How would one compute

```
(exists x (and (perfect-numberp x)
               (oddp x)))
```

Would you try all possible values of x? Even if you only tried all the natural numbers, they are infinite! (In this case, I'm looking to solve an unsolved problem in mathematics: the existence of an odd perfect number.)

So a nice thing about ACL2 formulas is that they are executable—if you assign a value to all the variables. For example, we could test our conjecture on there being no largest integer on the value 1000 like so:

```
(let ((x 1000))
  (implies (integerp x)
           (and (integerp (+ x 1))
```

```
(> (+ x 1) x))))
```

And that evaluates to T, meaning it is true.

Recall ACL2's notion of "Generalized Booleans," the notion of "true" and "false" used by IF? That ideas is used for "true" or "false" in formulas also. A formula that is always non-nil (in a given theory) is a theorem.

But wait… we can use any ACL2 expression as the test of an IF. The same is true for ACL2 formulas. The notion of generalized booleans allows any ACL2 expression to be interpreted as a formula. This means that the distinction between formulas and expressions we have in first order logic are not present in ACL2.

In first order logic, we also had a restriction that only formulas without free variables were propositions. But ACL2 is quantifier-free and free variables are implicitly universally quantified. Thus, all ACL2 formulas are propositions. Thus, all ACL2 expressions are propositions.

Here are some ACL2 theorems. They are theorems because there are no assignments to variables under which they evaluate to nil:

```
(< 1 2)

t

(integerp 5)

(implies (natp x) (integerp x))

5

"false"
```