MAT1830

Lecture 8: Predicate logic

"She is the most famous actor in the world."

is different from

"She is the most famous mathematician in the world."

"1 is the smallest positive integer."

is different from

"1 is the smallest real number."

8.1 Valid sentences

The language of predicate logic is based on predicate symbols, variables, constants, brackets, $\forall, \exists$ and connectives. The examples from last lecture illustrate how these ingredients are used to form sentences.

> A sentence in predicate logic is *valid* if it has value T under all interpretations.

This is similar to the definition of a tautology in propositional logic. But now "all interpretations" means all interpretations of the predicate symbols, which is more complicated. The interpretation of a symbol $P(n)$, say, must include both the range of the variable n, as well as saying those n for which $P(n)$ is true.

valid sentences in predicate logic

are analogous to

tautologies in propositional logic

Both are "always true"

...or more formally they are true under all interpretations.

But what we mean by an interpretation for predicate logic is more complicated than what we meant by an interpretation for propositional logic.

8.2 Interpretations

For example, one interpretation of $P(n)$ is "n is positive," where n ranges over the real numbers. Under this interpretation, $\forall n P(n)$ is false.

A different interpretation of $P(n)$ is "n is positive," where n ranges over the numbers > 2. Under this interpretation, $\forall n P(n)$ is true.

Unlike in propositional logic, there are infinitely many different interpretations of each formula. Thus there is no truth table method for predicate logic. We cannot decide whether a formula is valid by testing all interpretations.

$\forall x M(x)$

Two of the possible interpretations of this are:

$M(x)$ is "she is at least as famous as x" and x ranges over all actors.

$M(x)$ is "she is at least as famous as x" and x ranges over all mathematicians.

Question 8.1 Give interpretations which make the following false.

$\exists n P(n) \to \forall n P(n)$

n ranges over the positive integers and $P(n)$ is "n is prime"

$\forall x \forall y (R(x, y) \to R(y, x))$

x and y range over the real numbers and $R(x, y)$ is "$x < y$"

Flux Exercise (LQMTZZ)

Which of the following interpretations makes $\forall m \exists n S(m, n)$ false?

A. m and n range over the real numbers and $S(m, n)$ is "$m = n + 2$"

B. m and n range over the real numbers and $S(m, n)$ is "$m = 2n$"

C. m and n range over the integers and $S(m, n)$ is "$m = n + 2$"

D. m and n range over the integers and $S(m, n)$ is "$m = 2n$"

Answer: D.

8.3 Recognising valid sentences

Nevertheless, in some cases, we can see that a sentence *is* true for all interpretations.

Example. $\forall x \forall y P(x, y) \to \forall y \forall x P(x, y)$ is true for all properties P, and hence is valid.

Likewise, we can sometimes see that a sentence is *not* valid by finding an interpretation which makes it false.

Example. The sentence

$$\forall x \exists y Q(x, y) \to \exists x \forall y Q(x, y)$$

is false if we interpret $Q(x, y)$ as $x \leqslant y$ on the real numbers. With this interpretation

$$\forall x \exists y Q(x, y) \text{ is true}$$

(for any number there is a larger number), but

$$\exists x \forall y Q(x, y) \text{ is false}$$

(there is no number $\leqslant$ all numbers). Hence the implication is false.

8.4 Consequence and equivalence

As in propositional logic, a sentence ψ is a *logical consequence* of a sentence ϕ if any interpretation which makes ϕ true makes ψ true. Again we write $\phi \Rightarrow \psi$ if ψ is a consequence of ϕ, and this is the same as saying $\phi \to \psi$ is valid.

Example. Any interpretation which makes $\forall x \forall y P(x, y)$ true makes $\forall y \forall x P(x, y)$ true, and so $\forall x \forall y P(x, y) \Rightarrow \forall y \forall x P(x, y)$.

Similarly, sentences ψ and ϕ are equivalent, written $\psi \equiv \phi$, if each is a consequence of the other. Some sentences are equivalent for "propositional logic reasons."

Example. We have

$$\forall x(P(x) \wedge Q(x)) \equiv \forall x(Q(x) \wedge P(x))$$

simply because

$$P(x) \wedge Q(x) \equiv Q(x) \wedge P(x)$$

for any x.

However there are also equivalences that genuinely involve quantifiers.

Question 8.2 Give an interpretation to show that $\exists x(P(x) \wedge L(x))$ and $\exists x(P(x) \wedge \neg L(x))$ are not equivalent.

Let x range over the integers, and let $P(x)$ be "10 divides x" and $L(x)$ be "x is even".

Then $\exists x(P(x) \wedge L(x))$ is true because there is an integer that is divisible by 10 and is even (20 for example).

But $\exists x(P(x) \wedge \neg L(x))$ is false because there is no integer that is divisible by 10 and is odd.

So there is an interpretation in which one of these formulas is true and the other is false. This means they're not logically equivalent.

Question 8.3 Does $\forall x \exists y R(x, y) \Rightarrow \exists y \forall x R(x, y)$?

No. Let x and y range over the real numbers and $R(x, y)$ be $x < y$. Then $\forall x \exists y R(x, y)$ is true but $\exists y \forall x R(x, y)$ is false.

Question 8.4 Does $\exists y \forall x R(x, y) \Rightarrow \forall x \exists y R(x, y)$?

Yes. If $\exists y \forall x R(x, y)$ is true then there is one specific y, say y', such that $R(x, y') \equiv \mathsf{T}$ for all x. So then for any x there is a value of y, namely y', such that $R(x, y') \equiv \mathsf{T}$. So $\forall x \exists y R(x, y)$ is true.

Note that the argument above works for all possible interpretations of the sentences.

Question Does $\forall x P(x) \vee \forall x Q(x) \Rightarrow \forall x (P(x) \vee Q(x))$?

Thinking The LHS is effectively $(\forall x P(x)) \vee (\forall x Q(x))$. It's an $\vee$ of two separate $\forall$ statements. (It doesn't matter that the x's are the same). The RHS is a single $\forall$ statement.

Answer Yes. If the first statement is true, then $P(x)$ is true for all x in the range or $Q(x)$ is true for all x in the range. In each case this will mean that $P(x) \vee Q(x)$ is true for all x in the range, and so the second statement will be true.

Flux Exercise (LQMTZZ)

Does $\forall x(P(x) \vee Q(x)) \Rightarrow \forall x P(x) \vee \forall x Q(x)$? If not, give an interpretation which proves this.

A. No. x ranges over positive integers, $P(x)$ is $x \leq 10$ and $Q(x)$ is $x \geq 1$.

B. No. x ranges over positive integers, $P(x)$ is $x \leq 5$ and $Q(x)$ is $x \geq 15$.

C. No. x ranges over positive integers, $P(x)$ is $x \leq 10$ and $Q(x)$ is $x \geq 11$.

D. Yes.

Answer:

To show $\forall x(P(x) \vee Q(x)) \not\Rightarrow \forall x P(x) \vee \forall x Q(x)$ we need to make the LHS true and the RHS false.

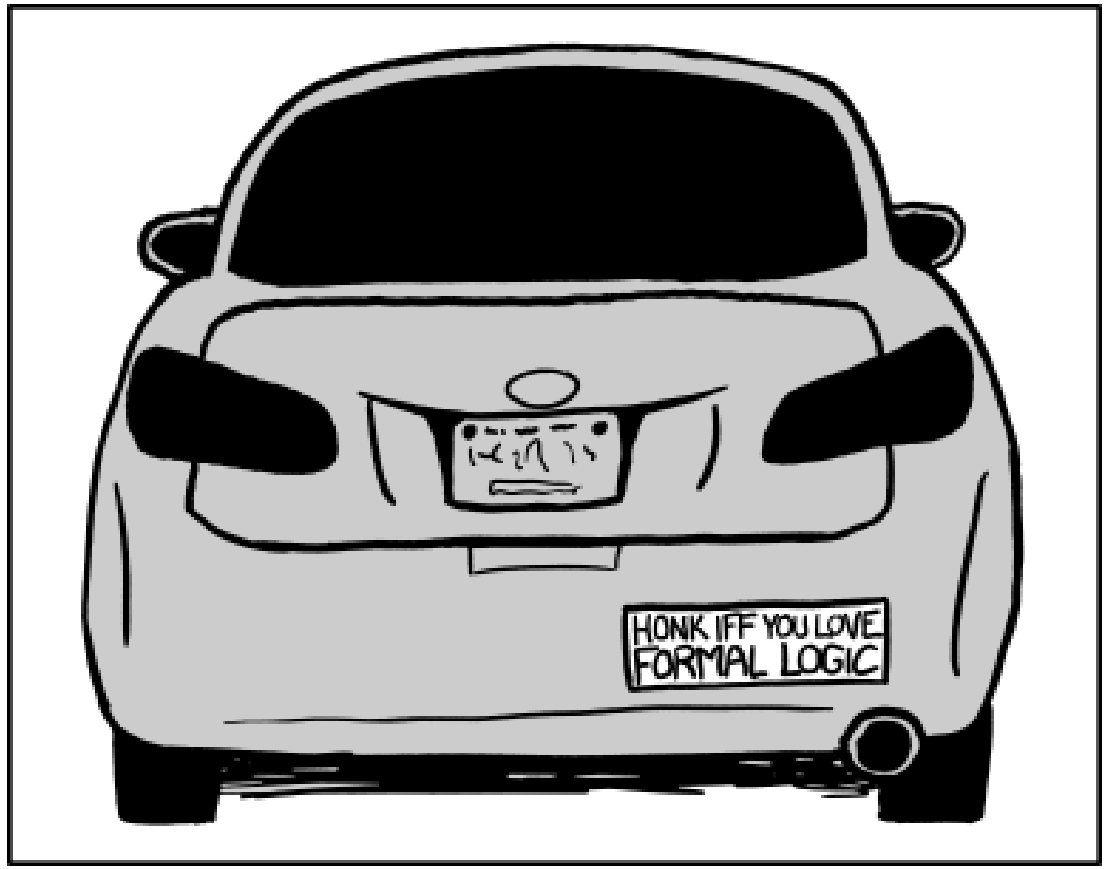
In A, both LHS and RHS are true.

In B, the LHS is false.

In C, the LHS is true and the RHS is false, as required.

D cannot be correct because C shows $\forall x(P(x) \vee Q(x)) \not\Rightarrow \forall x P(x) \vee \forall x Q(x)$

So C.


HONK IFF YOU LOVE
FORMAL LOGIC

8.5 Useful equivalences

Two important equivalences involving quantifiers are

$$\neg\forall x P(x) \equiv \exists x \neg P(x)$$

$$\neg\exists x P(x) \equiv \forall x \neg P(x)$$

These make sense intuitively. For example, $\neg\forall x P(x)$ means $P(x)$ is not true for all x, hence there is an x for which $P(x)$ is false, that is, $\exists x \neg P(x)$.

They can also be viewed as "infinite De Morgan's laws." If x ranges over $\{1, 2, 3, \ldots\}$ for example, then

$$\forall x P(x) \equiv P(1) \wedge P(2) \wedge P(3) \wedge \cdots$$

and

$$\exists x P(x) \equiv P(1) \vee P(2) \vee P(3) \vee \cdots$$

Hence

$$\begin{aligned}
\neg\forall x P(x) &\equiv \neg\left(P(1) \wedge P(2) \wedge P(3) \wedge \cdots\right) \\
&\equiv (\neg P(1)) \vee (\neg P(2)) \vee (\neg P(3)) \vee \cdots \\
&\quad \text{by de Morgan's law} \\
&\equiv \exists x \neg P(x).
\end{aligned}$$

And similarly

$$\begin{aligned}
\neg\exists x P(x) &\equiv \neg\left(P(1) \vee P(2) \vee P(3) \vee \cdots\right) \\
&\equiv (\neg P(1)) \wedge (\neg P(2)) \wedge (\neg P(3)) \wedge \cdots \\
&\quad \text{by de Morgan's law} \\
&\equiv \forall x \neg P(x).
\end{aligned}$$

Question 8.5 Explain why $\neg\forall p\forall t F(p,t) \equiv \exists p\exists t\neg F(p,t)$.

$$\begin{aligned} \neg\forall p\forall t F(p,t) &\equiv \exists p\neg\forall t F(p,t) \\ &\equiv \exists p\exists t\neg F(p,t) \end{aligned}$$

Intuitively: $\neg\forall p\forall t F(p,t)$ means it is not the case that $F(p,t)$ is true for all combinations of p and t. This is the same as saying that there is some combination of p and t for which $F(p,t)$ is false.

8.6 Simplification

The infinite de Morgan's laws allow a certain simplification of predicate formulas by "pushing $\neg$ inside quantifiers."

Example.

$$\begin{aligned} \neg\forall x\exists y Q(x,y) &\equiv \exists x\neg\exists y Q(x,y) \\ &\equiv \exists x\forall y\neg Q(x,y). \end{aligned}$$

It is in fact possible to transform any quantified statement in predicate logic to an equivalent with all quantifiers at the front.

8.7* Completeness and undecidability

In 1930, Gödel proved that there is a complete set of rules of inference for predicate logic. This means, in particular, that there is an algorithm to list the valid sentences.

However, in 1936, Church and Turing proved that there is no algorithm to list the logically false sentences. This means, in particular, that predicate logic is *undecidable*: there is no algorithm which, for any sentence ϕ, decides whether ϕ is valid or not.

This negative result is due to the power of predicate logic: it can express all mathematical or computational problems, and it is known that some of these problems cannot be solved by algorithm.

* this last part of the lecture is not assessable

Turing's halting problem (* not assessable)

Suppose that every program in a certain language either terminates or loops forever.

Question Is it possible to write a program which will accept the any piece of code in this language and determine whether the input code will terminate or loop forever?

If this was possible then we could easily write a program that would

- loop forever if the input code would terminate; and
- terminate if the input code would loop forever.

Now imagine feeding this program its own code. It will

- loop forever if it terminates; and
- terminate if it loops forever.

Which is nonsense.

Conclusion The answer to the question is no.