Utrecht University

Logic for Computer Science

05 – Predicate logic

Wouter Swierstra

University of Utrecht

Last time

Boolean algebra

Computer circuits

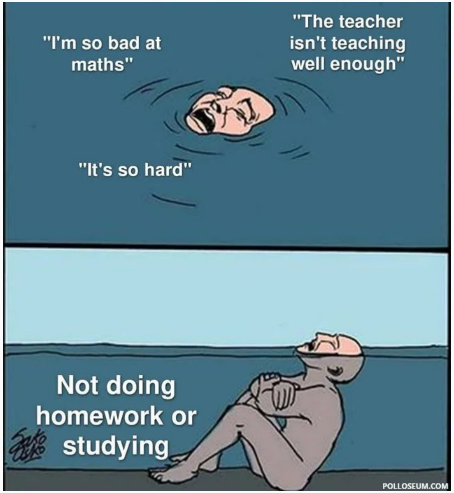
Binary arithmetic

This lecture

Predicate logic - Gottlob Frege

Organization

- Second quiz is today…
- Try to keep up – only a few weeks left to the mid-term.
- Not everyone comes to the exercise sessions…


"I'm so bad at maths"
"The teacher isn't teaching well enough"
"It's so hard"
Not doing homework or studying
POLLOSEUM.COM

A word of warning

Limits of propositional logic

When studying sets, we sometimes want to prove theorems in terms of the sets' elements.

Examples

- Two sets A and B are **disjoint** if there is no element $x \in A \cap B$.
- We say A is a **subset** of B when for all $x \in A$, we also have $x \in B$.

These statements make claims about elements of a set, but they are not simple propositions...

Socrates

Given the hypotheses:

- All men are mortal.
- Socrates is man.

Is it valid to deduce that ‘Socrates is mortal’?

Socrates

Given the hypotheses:

- All men are mortal.
- Socrates is man.

Is it valid to deduce that ‘Socrates is mortal’?

How do we know? We cannot model such statements using only propositional logic.

There is more going on in these statements than ‘just’ logical implication.

We need a *richer logic* to study such statements.

Predicates and propositions

Predicates

We sometimes defined sets by stating using the following notation:

- { x : x > 17}
- { p : p is a prime number}

Or more generally, we write {x : x has the property P}

Such a set consists of all elements that satisfy the **predicate** P.

In general, we will write P(x) when ‘x has the property P’, or when ‘P holds for x’.

Predicates versus propositions

A predicate is not the same as a proposition:

$P(x) = x > 17 \wedge x > 5$	defines a **predicate** on x
$3 < 12$	defines a **proposition**

You may want to think of a *predicate* as a function that computes a *proposition* for every value of the variable *x*.

More than one argument

Predicates can have more than one argument. In that case, they are typically called a **relation**.

We have already encountered several different relations when studying sets:

- SubsetOf(A,B) holds if for all x, $x \in A \Rightarrow x \in B$
- EqualSet(A,B) holds if both $A \subseteq B$ and $B \subseteq A$
- ProperSubset(A,B) holds if $A \subseteq B$ and $A \neq B$

Many familiar relations are written using infix operators, such as $\subseteq$ or =, rather than a function name, such as SubsetOf.

We'll see a lot more about relations in a few lectures...

Example: divisibility

We can 'define' a predicate $\mathit{Divides}(x, y)$ to hold when x and y are natural numbers and x divides y evenly (that is, there is no remainder after performing the division):

- For example, $\mathit{Divides}(3, 15)$ holds.
- But $\mathit{Divides}(3, 17)$ does not.

We can construct a truth-set:

$$\{(x, y) : \mathit{Divides}(x, y)\}$$

To be the set of all pairs (x, y) such that x divides evenly into y.

Traditionally, mathematicians write $x \mid y$ when $\mathit{Divides}(x, y)$ holds.

We'll see a more formal definition later today.

Predicates vs propositions – handling variables

Question

Is `c > 23` a predicate or a proposition?

Predicates vs propositions – handling variables

Question

Is `c > 23` a predicate or a proposition?

We can't say – are we defining a predicate on a variable c?

Or are we referring to some constant – like the speed of light?

As soon as we work with predicate logic, we need to treat variables extremely carefully.

Part of this lecture is aimed at introducing some associated terminology of studying variables and scoping.

Programming and predicates

If you write a program that contains undeclared variables, the compiler will typically reject your programming, saying that a variable is ‘not in scope’.

Programming and predicates

If you write a program that contains undeclared variables, the compiler will typically reject your programming, saying that a variable is ‘not in scope’.

If you write a proposition or predicate with variables, we have to be very careful about their meaning.

A predicate such as `x > 17` may hold for some values of `x`, but certainly not all.

Variables and substitution

Bound and free variables

When defining a predicate of the form:

```
P(x) = ...x...
```

The occurrences of `x` on the right hand side of the equality all refer to the `x` **bound** by the declaration `P(x)`.

If we write:

```
P(x) = ...y...
```

It is not clear what `y` is – we do not know where it is bound - we say that the variable `y` is **free**.

Example: substitution

We can turn any predicate into a proposition by **substituting** a value for variable bound in the predicate's definition.

For example, we can define the following predicate:

```
P(x) = x > 1337
```

- `P(10.000)` is the proposition `10.000 > 1337` (which happens to be true)
- `P(5)` is the proposition `5 > 1337` (which happens to be false).

Example: substitution

We can turn any predicate into a proposition by **substituting** a value for variable bound in the predicate's definition.

For example, we can define the following predicate:

```
P(x) = x > 1337
```

- `P(10.000)` is the proposition `10.000 > 1337` (which happens to be true)
- `P(5)` is the proposition `5 > 1337` (which happens to be false).

But we can also consider `P(y + 23)`, which corresponds to the propostion `(y + 23) > 1337`

The universe of discourse

Question

How many elements are there in the set $\{ x : x < 17\}$?

The universe of discourse

Question

How many elements are there in the set { x : x < 17}?

It depends!

Is it a set of natural numbers, integers, real numbers, ...

I prefer to be explicit:

$$\{x \in \mathbb{N} \;:\; x < 17\}$$

This avoids confusion and makes it clear what the *universe of discourse* is that I'm assuming.

These examples all show that – even in the study of formal logic – there can be information left implicit in the context, naming conventions, universe of discourse, etc.

Quantifiers

Repeated conjunction

Let A be the set {0,1,2,3}.

We say A is the subset of some other set B, written $A \subseteq B$, when **all** the elements of A also occur in the set B.

Or more precisely: $0 \in B \ \bigwedge \ 1 \in B \ \bigwedge \ 2 \in B \ \bigwedge \ 3 \in B$

Repeated conjunction

Let A be the set {0,1,2,3}.

We say A is the subset of some other set B, written $A \subseteq B$, when **all** the elements of A also occur in the set B.

Or more precisely: $0 \in B \wedge 1 \in B \wedge 2 \in B \wedge 3 \in B$

This may work for a **finite** set, but what if we want to show that all the even numbers are also natural numbers?

$0 \in \mathbb{N} \wedge 2 \in \mathbb{N} \wedge 4 \in \mathbb{N} \wedge 6 \in \mathbb{N} \wedge \ldots$

If we want to give a precise definition of a relation such as subsets, we need new notation and a more expressive logic.

Universal quantification

In **predicate logic** we can define the subset relation between A and B formally as follows:

$$\forall x \quad \left(x \in A \;\Rightarrow\; x \in B\right)$$

$\forall x \; P(x)$ is read as 'for all x, P holds for x'.

We call the 'upside down A' the **universal quantifier**.

Repeated disjunction

Two sets A and B are not disjoint if there is an element $x \in A \cap B$.

Let A be the set $\{0, 1, 2, 3\}$. When is A not disjoint from B?

$$(0 \in \mathrm{B} \ \bigvee \ 1 \in \mathrm{B} \ \bigvee \ 2 \in \mathrm{B} \ \bigvee \ 3 \in \mathrm{B})$$

Here we want to talk about repeated disjunction.

Repeated disjunction

Two sets A and B are not disjoint if there is an element $x \in A \cap B$.

Let A be the set $\{0, 1, 2, 3\}$. When is A not disjoint from B?

$$(0 \in B \ \bigvee \ 1 \in B \ \bigvee \ 2 \in B \ \bigvee \ 3 \in B)$$

Here we want to talk about repeated disjunction.

For this, we introduce the **existential quantifier**, written $\exists$.

We can formulate the proposition that A and B are not disjoint as:

$$\exists x \qquad (x \in A \ \bigwedge \ x \in B)$$

More generally, we write:

$$\exists x \qquad P(x)$$

Read as: there is some element x for which the predicate P holds.

Variations

Different textbooks use slightly different notation:

- $\forall x \in A \quad P(x)$ - making explicit that x is an element of some set A. This is sometimes referred to as **bounded quantification**.
- $\forall x.P(x)$ or $\forall x, P(x)$ - making clear where x ends and P starts. I'll use this on the slides occassionally: I find it makes formulas easier to read.
- $\forall x : Int32 \quad P(x)$ - making explicit that x has a certain *type*, such as Int32.
- And several other variations exist...

Examples

- Previously we 'defined' the Divides(x,y) relation to hold when x divides y evenly. Using quantifiers, we can give a more precise definition:

Divides(x,y) = $\exists k \quad k \times x = y$

- We can define the subset relation more precisely:

Subset(A,B) = $\forall x \quad (x \in A \Rightarrow x \in B)$

- Or the property that two sets are disjoint:

Disjoint(A,B) = $\neg\exists x \quad (x \in A \wedge x \in B)$

Example: GCD

Question

How would you formulate the proposition that *d* is the greatest common divisor of the two numbers *x* and *y*? You may use the Divides relation we saw on the previous slide.

Example: GCD

Question

How would you formulate the proposition that *d* is the greatest common divisor of the two numbers *x* and *y*? You may use the Divides relation we saw on the previous slide.

GCD(x,y,d) =

Divides(d,x) $\wedge$ Divides (d,y)

$\wedge$ $\forall$ c (Divides(c,x) $\wedge$ (Divides(c,y) $\Rightarrow$ c $\leq$ d)

(Other solutions may exist)

Variations

The existential quantifier, $\exists x \quad P(x)$, can be used to state that there is some x satisfying the predicate P.

It doesn’t say *how many different* x’s satisfy P.

Variations

The existential quantifier, $\exists x \quad P(x)$, can be used to state that there is some x satisfying the predicate P.

It doesn't say *how many different* x's satisfy P.

Sometimes people write:

$\exists! x \quad P(x)$

To mean there is **exactly one** x such that P(x) holds.

More than one quantifier

We can construct more complex formula's using more than one quantifier:

- $\forall x \forall y . x = y \Rightarrow y = x$
- $\forall x \exists y . y > x$
- $\exists x \in \mathbb{N} \forall y \in \mathbb{N} . x \leq y$
- $\forall x \exists y \exists z . x = y + z$

Question

What do these formulas mean?

Exactly one

We previously saw this variation of the existential quantifier:

$\exists!\, x \quad P(x)$

Question

Can we express this using the usual existential and/or universal quantifiers?

Exactly one

We previously saw this variation of the existential quantifier:

$\exists! x \quad P(x)$

Question

Can we express this using the usual existential and/or universal quantifiers?

There are several different ways to write this:

- $\exists x \quad (P(x) \wedge (\forall y \quad P(y) \Rightarrow x = y))$
- $\exists x \quad (P(x) \wedge (\neg\exists y \quad P(y) \wedge x \neq y))$
- $\exists x \quad (P(x) \wedge (\forall y \quad \neg P(y) \vee x = y))$
- ...

Conventions

In *Modelling computing systems*, $\forall x\ P(x) \Rightarrow Q(x)$ is to be read as $(\forall x\ P(x)) \Rightarrow Q(x)$.

Not all books agree however: other books interpret this formula as $\forall x\ (P(x) \Rightarrow Q(x))$.

I will try to always disambiguate between the two by explicit parentheses.

Multiple quantifiers

Does the order of quantifiers matter?

In other words, are $\forall x \exists y . P(x, y)$ and $\exists y \forall x . P(x, y)$ the same?

Multiple quantifiers

Does the order of quantifiers matter?

In other words, are $\forall x \exists y . P(x, y)$ and $\exists y \forall x . P(x, y)$ the same?

Let's consider the following example:

$\forall$ x $\exists$ y   x + y = 0

$\exists$ y $\forall$ x   x + y = 0

The first statement is true; the second is not. So the order of quantifiers really does matter.

Multiple quantifiers

$\forall x \forall y\, P(x,y)$ and $\forall y \forall x\, P(x,y)$ are equivalent.

Similarly, $\exists x \exists y\, P(x,y)$ and $\exists y \exists x\, P(x,y)$ are equivalent.

Question

Why is this?

Multiple quantifiers

$\forall x \forall y P(x,y)$ and $\forall y \forall x P(x,y)$ are equivalent.

Similarly, $\exists x \exists y P(x,y)$ and $\exists y \exists x P(x,y)$ are equivalent.

Question

Why is this?

A complete answer requires a formal study of predicate logic.

We'll cover this after Christmas.

But I can give some intuition for now.

Meaning of predicate logic

Brouwer-Heyting-Kolmogorov interpretation

What is a proof of some logical proposition P?

- a proof of $P \bigwedge Q$ consists of a proof of P and a proof of Q;
- a proof of $P \bigvee Q$ is either a proof of P or a proof of Q;
- a proof of $P \Rightarrow Q$ is a *function* that turns any proof of P into a proof of Q;
- a proof of $\neg P$ is a *function* that maps any proof of P into a proof of **F**;
- a proof of $\forall x \in A \quad P(x)$ is a *function* that, for each $a \in A$ computes a proof of $P(a)$;
- a proof of $\exists x \in A \quad P(x)$ consists of an element $a \in A$ and a proof that $P(a)$ holds;
- there is no proof of falsity.

Using the BHK interpretation

- A proof of $\exists x \in A \ \exists y \in B \ P(x,y)$ consists of:
  - a value $a \in A$
  - a value $b \in B$
  - a proof of $P(a,b)$
- A proof of $\exists y \in B \ \exists x \in A \ P(x,y)$
  - a value $b \in B$
  - a value $a \in A$
  - a proof of $P(a,b)$

Clearly we can convert between the two – hence they are equivalent.

Using the BHK interpretation

- A proof of $\forall x \in A\ \exists y \in B . P(x, y)$ consists of
  - a function that maps any $a \in A$ to a pair of a $b \in B$ and a proof of $P(a, b)$
- A proof of $\exists y \in B\ \forall x \in A . P(x, y)$ consists of:
  - an element $b \in B$
  - a function that maps every $a \in A$ to a proof that $P(a, b)$ holds.

Here we can see these proofs have a very different structure.

It is not at all obvious (and in fact impossible) to convert from the first to the second – how do we choose a $b \in B$?

The other conversion is possible, however.

Hence the implication holds in one direction, but not the other.

About the BHK interpretation

The BHK interpretation describes proofs in *intuitionistic logic*, a variant of the more popular *classical logic* that rejects certain axioms, such as:

- $p \bigvee \neg p$
- $\neg\neg p \Rightarrow p$
- ...

The consequence, however, is that our proofs become *executable* – each proof corresponds to a function that can be run on your computer.

It makes perfect sense for computer scientists to work in such a logic. Even if almost all mathematicians prefer classical logic, where the above axioms do hold.

As a result – depending on your logic - the BHK interpretation may suggest no proof exists, where there is a proof using classical axioms.

Binding and scope

Terminology about variable binding

Consider the following formula in predicate logic:

$$\forall x \quad P(x, y)$$

- the variable y is free;
- the quantifier $\forall$ *binds* the variable x, hence the occurrence of x in $P(x, y)$ is not free but **bound**.
- we can distinguish between the **binding occurrence** of x, namely $\forall x$, and the (regular) occurrences of the variable x (for example, as argument to the predicate P)

Scope

The **scope** of a quantifier is the part of a formula where the variable is bound:

$\forall x(P(x) \land Q(x))$

Here the scope of the universal quantifier is $(P(x) \land Q(x))$.

Scope

The **scope** of a quantifier is the part of a formula where the variable is bound:

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

Here the scope of the universal quantifier is $(P(x) \wedge Q(x))$.

When more than one quantifier binds the same variable, the occurrences of that variable refer to the nearest binding quantifier.

In this example, I’ve color-coded the variables to coincide with their corresponding quantifier:

$\forall x \quad (P(x) \wedge \exists x\, Q(x) \wedge R(x))$

Equivalent expressions

Are the following formulas in predicate logic equivalent?

- $\forall x \quad P(x)$
- $\forall y \quad P(y)$

Equivalent expressions

Are the following formulas in predicate logic equivalent?

- $\forall x \quad P(x)$
- $\forall y \quad P(y)$

They are ‘almost always’ equivalent.

Take $P(x)$ to be $\exists y \quad x \neq y$, for example. Then the two formulas correspond to:

- $\forall x \, \exists y \quad x \neq y$
- $\forall y \, \exists y \quad y \neq y$

Which are clearly not the same!

Equivalent expressions

We can freely *rename* bound variables, converting between

$\forall x \quad P(x)$ and $\forall y \quad P(y)$

Provided *x* and *y* do not occur freely in P.

This example highlights the kind of thing that can go wrong when working with variables.

Modelling in predicate logic

Modelling in predicate logic

Predicate logic is extremely useful when it comes to make natural language more precise.

We'll cover some (artificial) examples in the lecture today – but this is one of the key applications of logic in computer science.

When a customer comes up with a list of requirements, for example, translating these to (predicate) logic statements allows you to study them precisely and unambiguously – and perhaps even make clear why no solution exists.

Modelling in predicate logic

Every dog that has stayed in the kennel will have to go into quarantine.

Given predicates:

- K(x) = ‘x has stayed in the kennel’
- Q(x) = ‘x must go into quarantine’

We can express such a statement more precisely as:

$$\forall d \in Dog . K(d) \Rightarrow Q(d)$$

Ambiguity

Natural language is not very suitable for making these statement precise.

Given a predicate Loves(p,d) when a person p loves the dog d, written p ♡ d, what formula corresponds to:

Everybody loves a dog

Ambiguity

Natural language is not very suitable for making these statement precise.

Given a predicate Loves(p,d) when a person p loves the dog d, written $p \heartsuit d$, what formula corresponds to:

Everybody loves a dog

Two alternatives exist:

- $\exists\, d \in \text{Dogs}\ \forall\, p \in \text{People} .\ p \heartsuit d$
- $\forall\, p \in \text{People}\ \exists\, d \in \text{Dogs} .\ p \heartsuit d$

Without more information, it’s impossible to tell what the intended meaning is.

Modelling in predicate logic

- Everybody loves my baby
- But my baby don’t love anybody but me

Question

Write these statements as a formula in predicate logic.

Modelling in predicate logic

- Everybody loves my baby
- But my baby don't love anybody but me

Question

Write these statements as a formula in predicate logic.

We introduce two variables: b (for 'my baby') and m (for 'me') and a binary relation $\heartsuit$ (for 'loves).

- $\forall x \quad x \heartsuit b$
- $\forall x \quad b \heartsuit x \Rightarrow x = m$

Modelling in predicate logic

There are lots more examples in the exercises.

Translating between natural language and predicate logic is a great way to develop some intuition for predicate logic.

But beware: often many different (equivalent) solutions may exist!

Rules for quantification

Properties of quantifiers

We can generalize De Morgan's laws to work over quantifiers as follows:

- $\neg\forall x\, P(x) \quad \Leftrightarrow \quad \exists x\, \neg P(x)$
- $\neg\exists x\, P(x) \quad \Leftrightarrow \quad \forall x\, \neg P(x)$

We can 'prove' this by replacing the quantifiers with repeated conjunction/disjunction as necessary.

Properties of quantifiers

Similarly, we can show that quantifiers interact with conjunction/disjunction as follows:

- $\forall x\, (P(x) \wedge Q(x)) \quad \Leftrightarrow \quad (\forall x\, P(x)) \wedge (\forall x\, Q(x))$
- $\exists x\, (P(x) \vee Q(x)) \quad \Leftrightarrow \quad (\exists x\, P(x)) \vee (\exists x\, Q(x))$

Note: the other laws relating $\forall$-$\vee$ and $\exists$-$\wedge$ do **not** hold in general.

Proofs

We have already seen how to *prove* statements in propositional logic – by writing out a truth table.

Yet how can we prove statements in predicate logic?

To check if a statement of the form:

$$\forall\, n \in \mathbb{N} \quad P(n)$$

holds would require checking an *infinite* number of statements

Proofs

We have already seen how to *prove* statements in propositional logic – by writing out a truth table.

Yet how can we prove statements in predicate logic?

To check if a statement of the form:

$$\forall n \in \mathbb{N} \quad P(n)$$

holds would require checking an *infinite* number of statements

So how can we prove statements in predicate logic?

Proofs - example

We previously saw the pair of statements:

- Everybody loves my baby
- But my baby don’t love anybody but me

We introduce two variables: b (for ‘my baby’) and m (for ‘me’) and a binary relation $\heartsuit$ (for ’loves).

- $\forall x \quad x \heartsuit b$
- $\forall x \quad b \heartsuit x \Rightarrow x = m$

Question

What can we conclude about ‘my baby’ from these statements?

Proofs - example

We previously saw the pair of statements:

- Everybody loves my baby
- But my baby don’t love anybody but me

We introduce two variables: b (for ‘my baby’) and m (for ‘me’) and a binary relation ♡ (for ’loves).

- $\forall x \quad x \heartsuit b$
- $\forall x \quad b \heartsuit x \Rightarrow x = m$

Question

What can we conclude about ‘my baby’ from these statements?

From the first statement, we can conclude that $b \heartsuit b$.

But then we can use this – together with the second statement – to establish that $b = m$.

Hence ‘my baby’ is me.

Programming with predicates

Logic programming languages - notably Prolog - allow you to define custom predicates, stating that a list is sorted or a number is prime.

Rather than *execute* a program, you write a *query* of the form P(x)…

… and the execution engine then finds those values of x that satisfy the predicate P.

Programming with predicates

Logic programming languages - notably Prolog - allow you to define custom predicates, stating that a list is sorted or a number is prime.

Rather than *execute* a program, you write a *query* of the form `P(x)`...

... and the execution engine then finds those values of `x` that satisfy the predicate `P`.

Much more in the courses *Computationele intelligentie* and *Intelligente systemen*.

Proofs - perspective

In the next lecture, I'll give an 'intuitive' (but not formal) set of rules for writing proofs in propositional logic and predicate logic.

After the Christmas break, we'll study a formal notion of proof (for propositional logic).

But formalising the reasoning rules of predicate/propositional logic requires a bit more advanced mathematical machinery.

But the rules we'll cover next time should be enough to write out proofs like the one on the previous slide.

Recap

- Predicate logic: predicates, quantifiers, propositional logic operators
- Intuition: the BHK reading of predicate logic formulas
- Terminology for talking about variables: free, bound, scope
- How to model sentences from natural language in predicate logic
- Properties and equivalences between predicate logic formulas

But no **formal** notion of proofs just yet...

Material

- Modelling Computing Systems Chapter 4