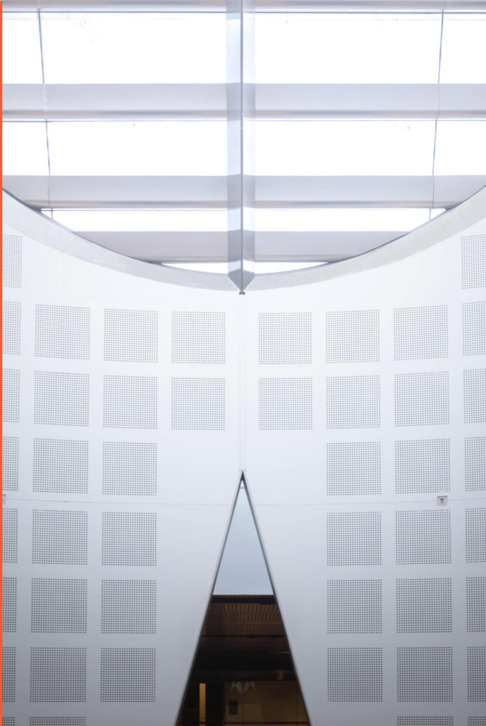
COMP2022|2922
Models of Computation

Introduction to Predicate Logic

Sasha Rubin

October 11, 2022

THE UNIVERSITY OF SYDNEY



Motivation for predicate logic

None of the following statements are propositions:

- "$x + 2$ is greater than 5".
- "$x + y$ is equal to $y + x$"

Why? Since whether or not it is true depends on missing information:

- the type (and values) of the variables x, y (integers? strings?)
- the meaning of the symbol $+$ (addition? concatenation?)

Predicate logic in a nutshell

Predicate-logic (aka First-order logic) is for modeling

1. objects (like numbers)
2. properties of objects ('x is even')
3. relations between objects (like 'x is greater than y').

It generalises Propositional Logic. The main new ingredients are:[1]

1. domain: collection of objects,
2. predicates: properties of objects and relations between objects.
3. variables: vary over objects in the domain.
4. quantifiers: allow one to reason about multiple objects.

[1] Often predicate logic also explicitly includes functions like $f(x) = x^2$, but we will not focus on these.

Domains

When we reason about objects, we have in mind a certain domain of discourse.

1. In programming, the domain may be the integers, or strings, or both, etc.
2. In the world, the domain includes people, animals, etc.

Definition

A domain is a non-empty set $\mathbb{D}$. It's elements are sometimes called objects.

- The set $\mathbb{Z}$ of integers is a domain. e.g., it contains -3.
- The set $\mathbb{H}$ of humans is a domain. e.g., it contains someone called `Romeo`.

Variables $x, y, z, \ldots$ vary over elements of the domain.

Predicates

In propositional logic we write propositions:

– 'Romeo is happy'
– 'Romeo loves Juliet'

In predicate logic we use predicates:

– $\texttt{happy}(\texttt{Romeo})$
– $\texttt{loves}(\texttt{Romeo}, \texttt{Juliet})$.

With variables we can also write atomic formulas:

– $\texttt{happy}(x)$
– $\texttt{loves}(x, y)$
– $\texttt{loves}(\texttt{Romeo}, y)$

Using connectives, we can now write some simple formulas:

– $\texttt{loves}(\texttt{Romeo}, \texttt{Juliet}) \rightarrow \texttt{happy}(\texttt{Juliet})$
– $\texttt{loves}(\texttt{Romeo}, x) \rightarrow \texttt{happy}(x)$

Predicates

Ok, but what is a predicate actually?

Definition

A predicate (of arity k) over domain $\mathbb{D}$ is a subset of $\mathbb{D}^k$.

- $\texttt{happy} \subseteq \mathbb{H}$ is a unary predicate $(k = 1)$.
- $\texttt{loves} \subseteq \mathbb{H} \times \mathbb{H}$ is a binary predicate $(k = 2)$.

We also allow infix notation:

$$x \,\texttt{loves}\, y$$

We also allow functional notation:

$$\texttt{loves} : \mathbb{H} \times \mathbb{H} \to \{\texttt{true}, \texttt{false}\}$$

Predicates

– Arguments in atomic formulas are variables and objects.

$$\texttt{loves}(x, \texttt{Juliet})$$

– If we fix the values of the variables, then predicates become propositions! And so are either true or false.

$$\texttt{loves}(\texttt{Romeo}, \texttt{Juliet})$$

– We cannot compose predicates...

$$\texttt{happy}(\texttt{loves(x,y)})$$

is not a formula and has no meaning.

Quantifiers

There are two types of quantifiers.

1. The existential quantifier, written $\exists$, read "exists"

$$\exists x F$$

   is true if **there is** an element d of the domain, F true when d replaces x in F.

2. The universal quantifier, written $\forall$, read "for all"

$$\forall x F$$

   is true if **for every** element d of the domain, F is true when d replaces x in F.

Quantifiers

Domain $\mathbb{Z}$ of integers

Predicates $\texttt{even}(x)$, $\texttt{odd}(x)$

Which of the following formulas are true?

Vote now! (on mentimeter)

1. $\exists x \, \texttt{even}(x)$
2. $\exists x \, (\texttt{even}(x) \wedge \texttt{odd}(x))$
3. $(\exists x \, \texttt{even}(x)) \wedge (\exists x \, \texttt{odd}(x))$

Quantifiers

Domain $\mathbb{Z}$ of integers

Predicates $\texttt{even}(x)$, $\texttt{odd}(x)$

Which of the following formulas are true?

Vote now! (on mentimeter)

1. $\forall x\, \texttt{even}(x)$
2. $\forall x\, (\texttt{even}(x) \vee \texttt{odd}(x))$
3. $(\forall x\, \texttt{even}(x)) \vee (\forall x\, \texttt{odd}(x))$

Quantifiers

A quantified formula

$$\exists x F$$

has two parts:

1. The variable being quantified x
2. The formula being quantified F.

We can nest quantifiers...

$$\exists x \exists y F$$

and even mix them...

$$\exists x \forall y F$$

Quantifiers

Domain $\mathbb{H}$ of humans

Predicate $\texttt{loves}(x, y)$

$\forall x\, \texttt{loves}(x, x)$	Every human loves themself
$\forall x \forall y\, \texttt{loves}(x, y)$	Everyone loves everyone
$\exists x \forall y\, \texttt{loves}(x, y)$	Someone loves everyone
$\forall x \exists y\, \texttt{loves}(x, y)$	Everyone loves someone
$\exists x \exists y\, \texttt{loves}(x, y)$	Someone loves someone

Translating to and from logic

- Think of logic as a programming language that is based in mathematics.
  - Programming languages (datalog, answer set programming)
  - Database query languages (SQL)
  - Hoare logic for verifying correctness of programs
- You should learn how to write formulas to say what you mean.

Translation Tips (i)

There are some common forms:

1. "All As are Bs" translates as $\forall x(A(x) \to B(x))$
2. "Some As are Bs" translates as $\exists x(A(x) \land B(x))$
3. "No As are Bs" translates as $\forall x(A(x) \to \neg B(x))$
4. "Some As are not Bs" translates as $\exists x(A(x) \land \neg B(x))$

Usually:

$\land$ goes with $\exists$

$\to$ goes with $\forall$

Translation Tips (i)

Domain $\mathbb{Z}$

Predicates $\texttt{even}, \texttt{odd}$ and $\texttt{greater}$

Translate the statement "Every even integer is greater than some odd integer" into predicate logic.

- This is of the form "All As are Bs"
- $A(x)$ for "x is an even integer"
- $B(x)$ for "x is greater than some odd integer"

$$\forall x(\texttt{even}(x) \to \exists y(\texttt{odd}(y) \land \texttt{greater}(x, y))$$

Translation Tips (i)

Translate the statement "Some even integer is equal to 0" into predicate logic.

Vote now! (on mentimeter)

1. $\exists x(\texttt{even}(x) \land \texttt{equal}(x, 0))$
2. $\exists x(\texttt{even}(x) \rightarrow \texttt{equal}(x, 0))$

Translation Tips (i)

Translate the statement "Every even integer is equal to 0" into predicate logic.

Vote now! (on mentimeter)

1. $\forall x(\texttt{even}(x) \wedge \texttt{equal}(x, 0))$
2. $\forall x(\texttt{even}(x) \rightarrow \texttt{equal}(x, 0))$

Translation Tips (ii)

The order of quantifiers only matters when mixing existential and universal quantifiers.

- $\forall x \forall y P(x, y)$ means the same thing as $\forall y \forall x P(x, y)$
- $\exists x \exists y P(x, y)$ means the same thing as $\exists y \exists x P(x, y)$
- $\exists y \forall x P(x, y)$ means there is a single y such that for all x we have that $P(x, y)$ is true.
- $\forall x \exists y P(x, y)$ means for every x there is a y (that can be different for different choices of x) such that $P(x, y)$ is true.

Translation Tips (ii)

Translate the following into logic in the domain of integers: "Every integer is greater than some integer"

Vote now! (on mentimeter)

1. $\forall x \exists y \, \texttt{greater}(x, y)$
2. $\exists y \forall x \, \texttt{greater}(x, y)$

Translation Tips (iii)

The formula $\forall x \neg P(x)$ is false when...?

Vote now! (on mentimeter)

1. $P(x)$ is true for every x.
2. $\neg P(x)$ is true for every x.
3. $P(x)$ is true for some x.
4. $\neg P(x)$ is true for some x.

Translation Tips (iii)

Which of the following is the negation of the formula

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

Vote now! (on mentimeter)

1. $\forall x \exists y \neg P(x, y)$
2. $\forall x \forall y \neg P(x, y)$
3. $\exists x \forall y \neg P(x, y)$
4. $\exists x \exists y \neg P(x, y)$
5. None of the above

Are there rules for manipulating formulas of predicate logic?

Equivalences involving quantifiers

For all formulas F, G:

$$
\begin{aligned}
&\text{(Q. Negation)} && \neg\forall x F \equiv \exists x \neg F \\
& && \neg\exists x F \equiv \forall x \neg F \\
&\text{(Q. Unification)} && (\forall x F \wedge \forall x G) \equiv \forall x (F \wedge G) \\
& && (\exists x F \vee \exists x G) \equiv \exists x (F \vee G) \\
&\text{(Q. Transposition)} && \forall x \forall y F \equiv \forall y \forall x F \\
& && \exists x \exists y F \equiv \exists y \exists x F \\
&\text{(Q. Extraction)} && \text{if } x \notin \mathrm{Free}(G) : \\
& && (\forall x F \wedge G) \equiv \forall x (F \wedge G) \\
& && (\forall x F \vee G) \equiv \forall x (F \vee G) \\
& && (\exists x F \wedge G) \equiv \exists x (F \wedge G) \\
& && (\exists x F \vee G) \equiv \exists x (F \vee G)
\end{aligned}
$$

Bound/Free variables

An occurrence of the variable x in the formula F is bound if x occurs within a subformula of F of the form $\exists x G$ or $\forall x G$. Otherwise it is a free occurrence.

- This is similar to local variables and global variables in programming.
- A variable may have both free and bound occurrences in a formula F.
- Intuitively, to get a proposition from a formula we need to instantiate all the free variables.
- A formula without free variables is called a sentence.

Self-test

Which variable occurrences are bound in the following formula?

$$\forall x\,(P(x,y) \rightarrow \exists y Q(x,y,z))$$

Vote now! (on mentimeter)

1. The x in P
2. The x in Q
3. The y in P
4. The y in Q
5. The z in Q

To give the precise syntax and semantics of predicate logic, we need to separate the vocabulary from the domain/structure.

- first-order structure
- vocabulary

A first-order structure (aka structure) consists of a domain $\mathbb{D}$, predicates on $\mathbb{D}$, and constants from $\mathbb{D}$.

e.g., $(\mathbb{Z}, \texttt{plus}, 0)$
e.g., $(\mathbb{S}, \texttt{plus}, 0)$

We will use superscripts to distinguish predicates.
e.g., $\texttt{plus}^{\mathbb{Z}}$ is addition on integers
e.g., $\texttt{plus}^{\mathbb{S}}$ is concatenation on strings[2]

[2]Similarly, $0^{\mathbb{Z}}$ is the integer 0, while $0^{\mathbb{S}}$ is the empty-string.

We distinguish between `plus` as a symbol, and as a relation on a specific domain.

A vocabulary (aka signature) is a collection of **symbols**, *i.e.*, predicate symbols and constant symbols (that are used as **names** for the predicates and constants in structures).

e.g., the vocabulary of both the structures $(\mathbb{Z}, \mathtt{plus}, 0)$ and $(\mathbb{S}, \mathtt{plus}, 0)$ have a single predicate `plus` and a single constant `0`.

Syntax of predicate logic

Fix a vocabulary.

Definition

A term is a variable $x, y, z, \ldots$ or a constant symbol $c, d, e, \ldots$.[3]

An atomic formula has the form $P(t_1, ..., t_k)$ where P is a k-ary predicate symbol and $t_1, ..., t_k$ are terms.

A formula is defined by the following recursive process:

1. Every atomic formula is a formula.
2. If F is a formula then $\neg F$ is a formula.
3. If F, G are formulas then $(F \wedge G)$ and $(F \vee G)$ are formulas.[4]
4. If F is a formula and x a variable then $\exists x F$ and $\forall x F$ are formulas.

---

[3] If we had included function symbols, then terms would also include things like $f(g(x))$ and $x^2 + 1$.

[4] Just as for propositional logic, we can also use other propositional connectives, e.g., $\rightarrow, \leftrightarrow$.

What about semantics of predicate logic?

– The truth value of a formula obviously depends on the structure it is interpreted over.
– But it also depends on the values of the variables.
– An assignment maps variables to objects.
  - Typically denote assignments by the letter α
  - So if F is a formula and α an assignment then $\alpha(F)$ is either true or false

Example

- Domain $\mathbb{Z}$
- Assignment $\alpha(x) = 3, \alpha(y) = 2$
- Which of the following formulas become true in this case?
  1. `greater_or_equal`(x, y) (aka $x \geq y$)
  2. `greater_or_equal`(y, y) (aka $y \geq y$)
  3. `greater_or_equal`(y, x) (aka $y \geq x$)

Example

- Domain $\mathbb{Z}$
- Formula $\forall x(x \geq y)$
- Assignment $\alpha(y) = 0$ (and we don't care what the value of α on x is, since x is bound).

The formula is false under the assignment. Why?

- Informally, the statement $\forall x(x \geq y)$ is true under α is the same as saying

  > "for every integer d, the formula $x \geq y$ is true under the assignment which agrees with α (on y) but maps x to d"

- And this statement is false, since we can take $d = -3$.
- On the next slide we will formalise this and give a recursive definition of semantics.

Semantics

Fix a vocabulary and a structure with domain $\mathbb{D}$.

Definition

The truth-value of a formula F under assignment α is defined recursively:

1. Predicates:
   1.1 Unary predicate P is true under α if $\alpha(x) \in P$.
   1.2 Binary predicate Q is true under α if $(\alpha(x), \alpha(y)) \in Q$.
2. The truth-value of the Boolean connectives are as usual.
3. $\forall x F$ is true under α if for every $d \in \mathbb{D}$, F is true under $\alpha[x := d]$.
4. $\exists x F$ is true under α if there is some $d \in \mathbb{D}$ such that F is true under $\alpha[x := d]$.

Here $\alpha[x := d]$ is the assignment that is identical to α except that it maps x to d. This is like replacing x by d.

Semantics

- Domain $\mathbb{Z}$
- Formula $\forall x(x \geq y)$
- Assignment $\alpha(y) = 0$ (and we don't care what the value of α on x is, since x is bound).

Let's apply the definition to show that the formula is false under the assignment.

- We want to know if $\forall x(x \geq y)$ is true under α.
- Same as $x \geq y$ being true under $\alpha[x := d]$, for every integer d.
- Same as $\alpha[x := d](x) \geq \alpha[x := d](y)$ for every integer d.
- Same as $d \geq 0$ for every integer d.
- This is false about the integers, e.g., take $d = -3$.
- Conclude that $\forall x(x \geq y)$ is false under α.

Validity

Fix a vocabulary.

A formula F is valid if it evaluates to true for every structure and assignment.

Examples

- $\forall x(P(x) \vee \neg P(x))$ is valid.
- $\forall x \exists y\, Q(y,x)$ is not valid since it is not true statement about the natural numbers with $Q(y,x)$ meaning $y < x$.

Validity

Does the following argument make logical sense?

1. All tall people are happy
   $\forall x(T(x) \to H(x))$
2. There is someone who is happy.
   $\exists x H(x)$
3. So, there is someone who is tall.
   $\exists x T(x)$

We can show that

$$(\forall x(T(x) \to H(x)) \wedge \exists x.H(x)) \to \exists x.T(x)$$

is not valid by finding a counterexample, i.e., domain and predicates that make it false.

	$T(x)$	$H(x)$
`Alan`	0	1
`Bob`	0	0

Validity

How do we show that a formula of predicate logic is valid?

- For propositional logic we could use truth-tables or deduction.

We will extend natural deduction to predicate logic.

- The new rules talk about quantifiers.
- ND will be sound and complete for predicate logic too.

Good to know

Is there an algorithm that decides if a given predicate logic sentence is valid?

- The language

  $$\{\langle F\rangle : F \text{ is a valid predicate-logic formula}\}$$

  is Turing-recognisable (since we have a sound and complete proof system!) but not Turing-decidable.
- Intuitively, this means that no algorithm can decide logical truth.

Good to know

Predicate logic can also include functions[5]

$$f : \mathbb{D}^k \to \mathbb{D}$$

For instance, in the domain of integers

$$\texttt{plus} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$$

$$\texttt{square} : \mathbb{Z} \to \mathbb{Z}$$

These can be written using infix notation.

Functions allow us to write terms that are more complex than simply variables and elements of the domain, e.g., $x^2 + 3$.

Terms can then be arguments in predicates, e.g., $\texttt{even}(x^2 + 3)$.

[5] We don't use them in this course.

More?

To learn more about predicate logic I recommend the following introductory texts:

1. Artificial Intelligence: A modern approach, Russell and Norvig, Chapter 8
2. Logic for Computer Scientists, Schöning, Chapter 2