Predicate Logic: the Need for a Richer Language

Some logical arguments cannot be expressed using propositional logic. A classical syllogism of Aristotle is

All humans are mortal
Socrates is human
Therefore, Socrates is mortal (11.1)

If we try to express this argument using propositional logic, we can see that propositions do not entirely capture our intent. For example, we might try rephrasing the syllogism to be

A human is mortal
Socrates is a human
Therefore, Socrates is mortal (11.2)

The logic of Argument 11.2 could be written as

\[ H \rightarrow M \]
\[ S \rightarrow H \]
\[ S \rightarrow M \] (11.3)

Although Argument 11.3 properly encodes Argument 11.2, it does not capture the sense of “all” that is in Argument 11.1¹.

To express such arguments we must be able to separate individuals from their properties. This is achieved by using predicates to describe properties or relationships. We will also introduce functions into the language to express mathematical relationships.

Roughly speaking, predicate logic is propositional logic augmented by quantifiers, predicates and functions. Many authoritative texts prefer to call this first-order logic, because only variables can be quantified; we will simply call it predicate logic.

¹This lack of expressiveness and rigor was observed by the philosopher Gottlob Frege in 1879, later made understandable and applied to number theory by Giuseppe Peano. We owe the systematic study of the foundations of mathematics to these and other intellectual pioneers.
We can begin to understand how predicate logic works by introducing variables into propositional logic. We might use the symbol \( x \) to represent a variable; for now we will not be concerned with values that a variable can have, leaving that until we discuss semantics.

One of the easier ways to represent the concept “A human is mortal” is to say that, if a variable satisfies the predicate “is human”, then it satisfies the predicate “is mortal. We can re-write the antecedent of the first line of Argument 11.2 as

\[
H(x) \rightarrow M(x)
\]

Introducing the predicate “is Socrates”, and writing that a variable satisfies this predicate as \( S(x) \), we can translate Argument 11.2 into a form that uses predicates:

\[
\begin{align*}
H(x) & \rightarrow M(x) \\
S(x) & \rightarrow H(x) \\
\hline
S(x) & \rightarrow M(x)
\end{align*}
\]  

(11.4)

This expresses the ideas in Argument 11.2 but it does not yet capture the concepts in the original syllogism of Argument 11.1. To do this, we need to quantify the variable \( x \). For the first line of the original syllogism, we could write

For all \( x \), if \( x \) is a human then \( x \) is mortal

This is typically done by using the universal quantifier symbol \( \forall \). The first line can be written, in predicate logic, as

\[
\forall x (H(x) \rightarrow M(x))
\]

The second line of Argument 11.1 is asserting that some individual is Socrates and that this individual is also human. There are at least two ways to write this, using implication and using conjunction. Choosing implication as better capturing the sense of the second line, we can write

There exists an \( x \) such that, if \( x \) is Socrates, then \( x \) is human

In predicate logic, this can be expressed using the existential quantifier \( \exists \) as the formula

\[
\exists x (S(x) \rightarrow H(x))
\]

To go further, we will formally define the language of predicate logic.

2The quantifier \( \exists \) was introduced by Peano in 1897; the quantifier \( \forall \) was introduced in 1935 by Gerhard Gentzen, who formalized natural logic as presented in this course.
Predicate Logic as a Formal Language

The axiomatic definition of predicate logic begins with new concepts that must be accepted and understood intuitively. These are concepts we already know from basic mathematics:

**Variable:** a thing that varies. It is a member of some set, which is the universe of discourse (what the logic discusses).

**Function:** a map from variables to a variable value. We will call a function of zero variables a nullary function; of one variable, a unary function; of two variables, a binary function; and so on, with a function of \( n \) variables called an \( n \)-ary function. The set of all functions is denoted as \( F \). From this, we specify a special kind of function:

**Constant:** A function of zero variables. This corresponds to a proposition.

**Predicate:** a thing that maps variables to truth values. When we get to semantics, we will see that a predicate is a set; for now, we will think of a predicate as a mapping from one or more variables to either \( T \) or \( F \). The space of all predicates is denoted as \( P \). In general, \( P \neq F \).

From these concepts and usages, we can define a **term** and a **formula**.

**Definition: Term**

- A variable is a term
- A constant is a term; a constant \( c \) is a nullary function in the space of all functions \( F \)
- If \( f \) is an \( n \)-ary function in \( F \) and \( t_1, \ldots, t_n \) are terms, then \( f(t_1, \ldots, t_n) \) is a term
- nothing else is a term

A variable is often written as a lower-case symbol from the “end” of the Latin alphabet, such as \( x, y \), etc. A constant is often written as a lower-case symbol from the “start” of the Latin alphabet, such as \( a, b \), etc. A function is often written as a lower-case symbol from the “late start” of the Latin alphabet, such as \( f, g \), etc.

Examples of terms:

\[
\begin{align*}
\text{a} & \quad \text{constant} \\
\text{x} & \quad \text{variable} \\
P(z) & \quad \text{predicate} \\
f(a, x) & \quad \text{function} \\
g(x, f(a, b), y) & \quad \text{function}
\end{align*}
\]

The last term is an example of how we can nest functions. This comes from the definition of a term, and how a function maps from variables to a variable-like value.
Axiomatically, we understand that if \( P \in \mathcal{P} \) is an \( n \)-ary predicate symbol, and \( t_1, t_2, \ldots, t_n \) are terms, then \( P(t_1, t_2, \ldots, t_n) \) expresses the relation \( P \) among the terms. We usually use the word “predicate” to indicate a unary function, and “relation” to indicate a binary function, but these usages are not definitions and are not strictly observed.

We have already discussed the quantifiers. For conciseness, these are

\[ \forall x \] is the symbolic representation of “for all \( x \)”

\[ \exists x \] is the symbolic representation of “there exists an \( x \)”

The syntactic specification of predicate logic is finalized by defining a formula to include everything in propositional logic, plus simple predicate and quantified predicates.

**Definition: Formula**

- If \( P \in \mathcal{P} \) is an \( n \)-ary predicate symbol, and \( t_1, t_2, \ldots, t_n \) are terms, then \( P(t_1, t_2, \ldots, t_n) \) is a formula
- If \( \phi \) is a formula, the \( (\neg \phi) \) is a formula
- If \( \phi \) and \( \psi \) are formulas, then \( (\phi \land \psi) \) and \( (\phi \lor \psi) \) and \( (\phi \rightarrow \psi) \) are formulas
- If \( \phi \) is a formula and \( x \) is a variable, then \( (\forall x \phi) \) is a formula and \( (\exists x \phi) \) is a formula
- Nothing else is a formula

The first three items in this definition capture the syntax of propositional logic. If we think of propositions, such as \( p \) or \( q \), as constant functions that are either \( \text{T} \) or \( \text{F} \), then we can see that they meet the syntactic and semantic requirements of predicate logic. The new concepts are in the fourth item, where quantification is defined.

The fourth item is often mis-understood and may need to be the subject of further study by some students. For example, the symbolic string

\[ (\forall x P(x, y)) \]

is a formula but the variable \( y \) is not “mentioned” in the quantifier. This is still a formula! We will discuss this later in the course when we get to the *scope* of a quantifier. For now, as computer scientists, we understand that the variable \( y \) is “global” to the formula.
Practice Problems for Predicates

Many students need practice in translating natural language into symbolic logic. These are some problems of the type that students are expected to be able to translate.

Translations

Using these predicates, and the nullary or constant function $m$,

- $A(x, y): x$ admires $y$
- $B(x, y): x$ attended $y$
- $P(x): x$ is a professor
- $S(x): x$ is a student
- $L(x): x$ is a lecture
- $m$: is Mary

Translate the following English sentences into predicate logic:

- Mary admires every professor.
- Some professor admires Mary.
- Mary admires herself.
- No student attended every lecture.
- No lecture was attended by every student.
- No lecture was attended by any student.