We can think of the syntax of a formal language as an answer to the question, “what are we allowed to write?” We can think of the rules of deduction, variously, as answers to a question:

- From premises, how can we deduce a conclusion?
- How can we transform the premise sentences into a conclusion?

The second way is very important to a computer scientist because a great deal of what we do is transforming one representation into another.

Semantics are an answer to the simple question, “does this make sense?” By this we mean that, if we assign meanings to variables and functions and predicates, does a proof preserve the truth properties through the allowed syntactic transformations?

To explore the semantics of predicate logic, let us recall the semantics of propositional logic and how we extended its syntax.

The foundation of propositional semantics was that every “atom” had the value \( T \) or the value \( F \). From this, we introduced a model, which was an assignment of truth values to the atoms. We expressed a model, primarily, as a truth table. We discovered two critically important properties of a truth table: a proposition was valid when every row of the truth table was \( T \), and a proposition was satisfiable when some row of the truth table was \( T \). (We really referred to a model rather than to a row of a truth table, but for now the truth table will suffice.)

An immediate difficulty we spotted, when a proposition had \( k \) atoms, was that the truth table had \( 2^k \) rows. It grew exponentially, leaving us with an uneasy feeling about evaluating validity and satisfiability of large propositions. We will have to face this uneasiness directly now, because when we try to think of a model in predicate logic we have to manage the possibility that a variable can take on one of an arbitrarily large number of values.

Our semantics begins with a need to restrict the values that a variable or function can have. Many texts call this the universe of discourse, which is the set of values that we are talking about; this course’s textbook uses similar language, calling it the universe of concrete values. In an axiomatic system we must take this concept as intuitively understood.

We will refer to the universe of discourse as the set \( A \). Following Georg Cantor, we will think of a set as a collection of “clearly defined, distinguishable objects”\(^1\).

\(^1\)Cantor wrote in German with, surprisingly, a definitive citation not yet clear. One translation is from a footnote on Page 679 of *Mathematical Statistics for Economics and Business*, by R. Mittelhammer (1995) which is: “By set we mean any collection \( M \) of clearly defined, distinguishable objects \( m \) (which will be called elements of \( M \)) which from our perspective or through our reasoning we understand to be a whole”.

---

77
A variable, such as \( x \) or \( y \), will be restricted to taking on a value from the universe of discourse. In mathematical notation, we would write

\[
x \in A
\]

Next, we need to understand the concept of a mapping, which we will abbreviate as a map. This “takes” its inputs “to” its outputs. Here, a map will always have a single determinable result for any given inputs.

An important concept is that a predicate \( P \) can be either a mapping or a set. As a mapping, a predicate takes one or more input values and returns either \( T \) or \( F \). In mathematical notation, we would write the mapping concept as

\[
P : A \rightarrow \{ T, F \}
\]

We would write the set concept as

\[
x \in P
\]

The set of all predicate symbols will be written as \( \mathcal{P} \), so a predicate \( P \) must satisfy the set equation \( P \in \mathcal{P} \).

We can now consider some simple examples of a set \( A \) and predicates \( P \) and \( Q \). Suppose we restrict our values to be the integers from 0 to 3; we would then have

\[
A = \{0, 1, 2, 3\}
\]

We might define \( P(x) \) to be “\( x \) is even” and \( Q(x) \) to be “\( x \) is odd”. With this specific universe \( A \), and these predicates \( P \) and \( Q \), we can intuitively see that certain assertions are always true:

\[
\begin{align*}
\exists x \ P(x) \\
\exists x \ Q(x) \\
\forall x \ (P(x) \lor Q(x))
\end{align*}
\]

It is crucially important that we understand that these assertions are not tautologies, because we cannot prove them as theorems in predicate logic; but they are definitely true for the specific set \( A \) and meanings \( P \) and \( Q \) that we just used. To extend the concepts from propositional logic: these turn out to not be valid formulas, but they are satisfiable.

The next concept we need is that of a function. Syntactically, we simply accepted that a function “takes” zero or more terms as input and “returns” a value that a variable can take. We now need to be more careful and specific.

With reference to a set \( A \), we want a nullary (or constant) function to always return a value that is in the set \( A \). In mathematical notation, we would write

\[
f : \{} \rightarrow A
\]

This notation says that, taking no inputs, the function \( f \) maps to the set \( A \); this is the same as saying that it returns a value in the set \( A \).
Now, suppose that \( f \) is a function that maps a single value to one definite value. In mathematical notation, we would write
\[
f : A \rightarrow A
\]
Next, suppose that \( f \) maps two values to one definite value. We write this as \( f(x, y) \) according to the syntax of propositional logic. Semantically, we mean that \( f(\cdot, \cdot) \) maps a pair, or 2-tuple, of values from the set \( A \) to a value in the set \( A \). In mathematical notation, there are two ways to write this; one way is to say explicitly that \( f \) takes a 2-tuple from the outer-product set \( A \times A \), and the other way is to abbreviate the outer-product set as \( A^2 \). The instructor’s preferred notation, and the textbook notation, are
\[
\begin{align*}
& f : A \times A \rightarrow A \\
\text{or} \quad & f : A^2 \rightarrow A \\
\text{equivalently,} \quad & z = f(x, y) \Rightarrow z \in A
\end{align*}
\]
This notation extends to a function of \( n \) variables, also called an \( n \)-ary function, as
\[
\begin{align*}
& f : A \times A \times \cdots \times A \rightarrow A \\
\text{or} \quad & f : A^n \rightarrow A
\end{align*}
\]
We must use our notation carefully. In symbolic logic, the set of functions that we are using is typically limited; the set is written using the symbol \( \mathcal{F} \). When we mean that a function \( f \) is in the set of specified functions, we will write
\[
f \in \mathcal{F}
\]
This is very different than the usual way of writing function! We might be used to saying that the value returned by \( f \) is in the set \( A \); we do not need to say this, because it is in the definition of the function \( f \). Instead, we are concerned with restricting our attention to one of a limited number of functions.

For example, thinking of the previous set \( A \) that is the integers from 0 to 3, we might define the function \( f \) to be the “next” integer in base-4 arithmetic. This is often written as modular arithmetic; we could define the function \( f \) in many ways, such as
\[
\begin{align*}
f(x) & \overset{\text{df}}{=} (x + 1) \mod 4 \\
f(x) & \overset{\text{df}}{=} \begin{cases} 
  x + 1 & \text{if } x \leq 2 \\
  0 & \text{if } x = 3
\end{cases}
\end{align*}
\]
We might define the function \( g \) to be “double” the integer in base-4 arithmetic, so
\[
g(x) \overset{\text{df}}{=} (2 \cdot x) \mod 4
\]
Our set, or space, of functions would then be \( \mathcal{F} = \{ f, g \} \).
We now have the concepts needed to construct a model in predicate logic. We will use the symbol $\mathcal{M}$ to refer to a model. Before getting to the definition, let us be sure that we are careful about what we mean.

A model $\mathcal{M}$ needs to specify, at minimum: a universe of discourse $A$; a set of functions $\mathcal{F}$; and a set of predicates $\mathcal{P}$. The model must also be able to “ground” any constants, and be able to compute each function and each predicate.

Within a model $\mathcal{M}$, a function $f$ or a predicate $P$ has an interpretation; this is, simply, what the function or predicate means within the model. We will write this interpretation as $f^M$ and $P^M$, respectively.

For example, the interpretation of the “next” function will differ for modulo 4 arithmetic and modulo 8 arithmetic; if modulo 4 is model $\mathcal{M}$, and modulo 8 is model $\mathcal{M'}$, then the two interpretations of $f$ are $f^M$ and $f^{M'}$. We will write the superscripts so that we are completely clear and specific, but within a model we will understand the interpretations of functions and predicates.

**Definition:** A model $\mathcal{M}$ for predicate logic.

Let $\mathcal{F}$ be a set of function symbols and $\mathcal{P}$ a set of predicate symbols, each symbol with a fixed number of required arguments. A model $\mathcal{M}$ of the pair $(\mathcal{F}, \mathcal{P})$ consists of the following set of data:

1. A non-empty set $A$, the universe of concrete values;
2. for each nullary function symbol $f \in \mathcal{F}$, a concrete element $f^M$ of $A$;
3. for each $f \in \mathcal{F}$ with arity $n > 0$, a concrete function $f^M : A^n \to A$;
4. for each $P \in \mathcal{P}$ with arity $n > 0$, a subset $P^M \subseteq A^n$ of $n$-tuples over $A$.

We already have an example of a model, which is part of base-4 arithmetic. Let us build that part into a model for predicate logic.
Example: Part of base-4 arithmetic

We previously specified the set $A$, a successor function $f$, and two predicates $P$ and $Q$ for even and odd integers. Let us add a constant for the “initial” integer, which is zero; we can use the symbol $i$ for this initial integer. Our model, so far, would look like:

$$A \overset{df}{=} \{0, 1, 2, 3\}$$
$$P \overset{df}{=} \{P, Q\}$$
$$F \overset{df}{=} \{f, i\}$$
$$P^M \overset{df}{=} \{0, 2\}$$
$$Q^M \overset{df}{=} \{1, 3\}$$
$$i^M \overset{df}{=} 0$$
$$f^M(x) \overset{df}{=} (x + 1) \mod 4$$

This way of writing the predicate $P$ and $Q$ differs considerably from how we usually write a predicate, so let us consider what the textbook notation means. The idea for the predicate $P$ in our model $M$ is that, if a term $t$ evaluates to either 0 or 2, then $P$ is $T$; this is the same as saying that $t \in \{0, 2\}$. Otherwise, $P$ is $F$. Similarly for the predicate $Q$ in our model $M$, if a term $t$ evaluates to either 1 or 3, then $Q$ is $T$; this is the same as saying that $t \in \{1, 3\}$. Otherwise, $Q$ is $F$.

Within this model $M$, we can verify that these formulas are true:

$$P^M(i^M)$$
$$\neg Q^M(i^M)$$
$$\neg P^M(f^M(i^M))$$

Extra Notes

The definitions of predicates that we have provided so far in this course are intensional, where we have described the necessary and sufficient conditions for a predicate to be true. An equally legitimate definition of a predicate is extensional, where we enumerate or otherwise specify the arguments for which the predicate is true. This is especially useful for small finite sets, but can be used for larger sets. The choice of an intensional or extensional definition will depend on circumstances. For the semantics of predicate logic, we will always prefer the extensional definition.

In the above example of base-4 arithmetic, we used an extensional definition of the predicates $P^M$ and $Q^M$. The English description that follows the definition also provides an intensional definition of each predicate.

End of Extra Notes