CISC 204 Class 25

Models of Predicate Logic

Text Correspondence: pp. 124–127

Main Concepts:

- *Model: semantics of predicate logic described using sets*
- *Equality: has fixed semantics in all models*

We now have the concepts needed to construct a *model* in predicate logic.

25.1 Models of 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 is an explicit mapping; this is, simply, what the function or predicate means within the model. We will write these explicit models as $f^{\mathcal{M}}$ and $P^{\mathcal{M}}$, respectively.

For example, the model 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 models of f are $f^{\mathcal{M}}$ and $f^{\mathcal{M}'}$. We will write the superscripts so that we are completely clear and specific, but within a model we will understand the models of functions and predicates.

© R E Ellis 2022

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 $f^{\mathcal{M}}$ of A;
3. for each $f \in \mathcal{F}$ with arity $n > 0$, a concrete function $f^{\mathcal{M}} : A^n \to A$;
4. for each $P \in \mathcal{P}$ with arity $n > 0$, a subset $P^{\mathcal{M}} \subseteq A^n$ of n-tuples over A.

The definition of a model is important and can be difficult to fully understand. Let us go through some examples of models so that we can see some applications of predicate logic. We already have an example of a model, which is part of base-4 arithmetic, so we can that part of arithmetic into a model for predicate logic.

Observation: semantics of the equality predicate "="

There is one predicate that always has a fixed semantics in every possible model. The predicate is "=", for equality of terms, and its semantics must be specified for the $=$ i and $=$ e rules of deduction to operate properly.

The rules of deduction for equality are, in effect, a *intensional* definition of "=". These rules specify how terms can be syntactically replaced, and that a term can be replaced with itself. Together, these rules describe the intent of equality and how it operates on terms.

The semantics of equality are simple to specify with an *extensional* definition.

Definition: semantics of equality

For a set A in a model $\mathcal{M}$, for two values $a \in A$ and $b \in A$,

$(a, b) \in =^{\mathcal{M}}$ means that a and b are the same element in the set A. This can be written as $a =^{\mathcal{M}} b$.

Because the semantics of equality are fixed, we can omit the mention of a model and always write $a = b$ with no possibility of confusion. This puts our common usage on a solid semantics foundation.

The fixed semantics of equality also makes the evaluation of a model easier, because set theory and searching can be employed in the process of evaluating a predicate formula.

© R E Ellis 2022

Example: Part of base-4 arithmetic

We previously specified the set A, a successor function s, 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:

$$\begin{array}{rcl} A & \stackrel{\text{def}}{=} & \{0,1,2,3\} \\ \mathcal{P} & \stackrel{\text{def}}{=} & \{P,Q\} \\ \mathcal{F} & \stackrel{\text{def}}{=} & \{s,i\} \\ P^{\mathcal{M}} & \stackrel{\text{def}}{=} & \{0,2\} \\ Q^{\mathcal{M}} & \stackrel{\text{def}}{=} & \{1,3\} \\ i^{\mathcal{M}} & \stackrel{\text{def}}{=} & 0 \\ s^{\mathcal{M}}(x) & \stackrel{\text{def}}{=} & (x+1)\, mod\, 4 \end{array}$$

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 $\mathcal{M}$ 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 $\mathcal{M}$, 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 $\mathcal{M}$, we can verify that these formulas are true:

$P^{\mathcal{M}}(i^{\mathcal{M}})$

$\neg\, Q^{\mathcal{M}}(i^{\mathcal{M}})$

$\neg\, P^{\mathcal{M}}(s^{\mathcal{M}}(i^{\mathcal{M}}))$

The equality predicate, so far in this course, has been used only for substitution of terms. Within the model of base-4 arithmetic, we can use equality to make assertions.

We might, for example, want to show that the function f "wraps around". We can assert equality between the values in A and the results of applying f as:

Explicit	**Understood in $\mathcal{M}$**
$1 = s^{\mathcal{M}}(i^{\mathcal{M}})$	$s(i)$
$2 = s^{\mathcal{M}}(s^{\mathcal{M}}(i^{\mathcal{M}}))$	$s(s(i))$
$3 = s^{\mathcal{M}}(s^{\mathcal{M}}(s^{\mathcal{M}}(i^{\mathcal{M}})))$	$s(s(s(i)))$
$0 = s^{\mathcal{M}}(s^{\mathcal{M}}(s^{\mathcal{M}}(s^{\mathcal{M}}(i^{\mathcal{M}}))))$	$s(s(s(s(i))))$

We can also make assertions that use "=" as a predicate. An example of a formula ϕ that holds in this model $\mathcal{M}$ is

$$\forall x\, ((x = f(i)) \rightarrow Q(x))$$

© R E Ellis 2022

As a matter of technical detail, we usually do not specify a model in a quantified formula. We are often concerned with questions such as "is $\forall x\, \phi$ true in all models?" or "is there a model for which $\neg\exists x\, \psi$?" We then examine the quantified formula within a model of interest.

Example: Binary strings

The textbook, in Example 2.16, has a universe of discourse that is infinite. The set A in its example is the set of all binary strings, that is, strings composed entirely of 0's and 1's; the empty string ϵ is a string, which "grounds" the set. An element of A, for this example, will be called a *word*.

We can "build" the set A by recursive definition, much as we did when we defined terms in predicate logic. One way to do this is by saying that the empty string is a word in A. We can then say that, if w is a word in A, then the strings $w0$ and $w1$ are also words in the set A. Formally, the set can be defined as

$$\epsilon \in A$$
$$w \in A \rightarrow w0 \in A$$
$$w \in A \rightarrow w1 \in A$$

We could also have built A by putting the 0's and 1's "ahead" of existing words, but we will stick with this definition for now.

Let $\mathcal{F} \stackrel{\text{def}}{=} \{e, \cdot\}$ where e is a nullary constant function and $\cdot$ is a binary function; instead of $\cdot(x, y)$ we will write $x \cdot y$ using "infix" notation. Let $\mathcal{P} \stackrel{\text{def}}{=} \{\leq\}$ where $\leq$ is a binary predicate; instead of $\leq (x, y)$ we will write $x \leq y$, again using "infix" notation. An example of usage is, for terms t_1 and t_2, the formula $(t_1 \cdot t_2) \leq (t_2 \cdot t_1)$; for strings, this formula is more convenient than the "prefix" notation $\leq (\cdot(t_1, t_2), \cdot(t_2, t_1))$ but the two notations have the same semantics.

This defines the universe of discourse, the set of predicates, and the set of functions. Next, we must provide a model of the function $e^{\mathcal{M}}$, the function $\cdot^{\mathcal{M}}$, and the predicate $\leq^{\mathcal{M}}$.

Model $\mathcal{M}$ has as its universe A all finite binary strings, that is, all finite strings over the alphabet $\{0, 1\}$. The constant e denotes the empty word, so

$$e^{\mathcal{M}} \stackrel{\text{def}}{=} \epsilon = \{\}$$

The function $\cdot^{\mathcal{M}}$ is modeled as the string that is the concatenation of the two words that are its arguments. For example,

$$0011 \cdot^{\mathcal{M}} 101 \text{ equals } 0011101$$

To be more formal, we might state this as

© R E Ellis 2022

For $s_1 \in A$ and $s_2 \in A$, the concatenation $s_1 \cdot^{\mathcal{M}} s_2$ is the string $s_1 s_2$

The authors model $\leq^{\mathcal{M}}$ as a prefix ordering on binary strings. This means that the predicate $\leq^{\mathcal{M}}$ is true if and only if the first argument is a prefix of the second argument. An example is

$$011 \leq^{\mathcal{M}} 01101$$

A counter-example is

$$\neg\,(001 \leq^{\mathcal{M}} 01101)$$

The textbook provides an extensional definition of the predicate $\leq^{\mathcal{M}}$. One intensional definition of $\leq^{\mathcal{M}}$ is that we mean it as

For $s_1 \in A$ and $s_2 \in A$,

$s1 \leq^{\mathcal{M}} s_2$ if and only if there exists $s_3 \in A$ such that $s_1 s_3 = s_2$

An extensional definition, defining $\leq^{\mathcal{M}}$ as a set, is

$$\leq^{\mathcal{M}} \stackrel{\text{def}}{=} \{(s_1, s_2) | \exists_{s_3 \in A}((s_1 \cdot^{\mathcal{M}} s_3) = s_2)$$

© R E Ellis 2022