CISC 204   Class 27

Logical Environments

Text Correspondence: pp. 126–127

Main Concepts:

- *Environment: a model that assigns each unbound variable to a value*
- *Lookup table: tabular specification of unbound variables*

The main problem we face now, when we try to extend the semantics of a truth table to predicate logic, is how to consistently manage variables.

27.1 Unbound Variables

If we have just one predicate in a formula, such as $\exists x\, P(x)$, we can imagine a process for assigning each value in the universe to x until we find one that has the property P. Trying to do this for a more complicated formula turns out to require some logical machinery.

We have resolved this difficulty in handling variables by turning from intensional definitions of predicate and instead use *extensional* definitions. This changed our point of view and, for a computer, means that determining whether a predicate holds will be the process of determining whether a given thing is in a *set*.

We saw this in the example of a model for a finite-state machine. The relation R was defined extensionally as a set, so that determining whether the relation held between two elements of the universe A was a matter of determining whether a given pair was in the set R.

This is the insight of extensional definition: *a predicate is a set*. For an intensionally defined predicate $\underline{P}^{\mathcal{M}}$ in a model $\mathcal{M}$, we can give the extensional definition

$$P^{\mathcal{M}} \overset{\text{def}}{=} \{x | \underline{P}^{\mathcal{M}}(x)\} \tag{27.1}$$

We can extensionally define a predicate by a set rule, or by enumeration, according to the application of interest.

For example, there are many equivalent definitions of the set of odd integers. Intensionally, we could say that an odd integer is an integer that passes a test of not being divisible by 2. This might be something like

$$\underline{P}^{\mathcal{M}}(x) : x \equiv 1 \bmod 2$$

Extensionally, we could say that the set of odd integers is the set of numbers that are the results of adding 1 to an even number, which could be

$$P^{\mathcal{M}} = \{x | x = (2k+1) \text{ for some } k\}$$

or the set of all integers minus the set of all even integers, which could be

$$P^{\mathcal{M}} = \mathbb{Z} \setminus 2\mathbb{N}$$

From now on, we will assume that any predicate or relation is a set.

To be able to properly describe predicates, we must be able to handle variables. For example, consider the implication

$$P(c) \rightarrow \exists x\, P(x)$$

If we have a model $\mathcal{M}$, and it has the constant $c \in A$, we can write the antecedent as

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

When we try to model the consequent, we observe that the variable x is free in the formula $P(x)$. We can model the predicate, because it works out to a set by an extensional definition, but the symbolic string

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

does not make sense. The key question this raises is, "What is a model for a variable?"

To be able to properly handle variables, we introduce the concept of an *environment*. We already have this concept from computing, where the environment is the context within which a computation happens; it is the hardware plus the software. In predicate logic, the environment is the set of variables and how to map the variables to the universe of discourse A. We will formalize an environment as a mapping, which has two parts.

We will use the symbol l to represent an environment. The domain of the mapping l is the set of valid inputs to the map, which is an extensional definition. In mathematics this set would be written as "dom l" but in our textbook the set is written as var.

The set var is the name of all variables of a set of formulas. We can see, immediately, that the environment and the formulas are closely related; this makes sense, because in computing when we declare a new global variable we are adding it to the environment. For now, let us suppose that an environment can model all of the variables in our formulas of interest.

An environment is an extensionally defined mapping. By this, we mean that an environment l is a set of tuples. The first element of a tuple is a variable name and the second element of a tuple is a value from the universe of discourse A. For example, if we have two variables y and z, and the universe is three values $A = \{a, b, c\}$, then we could have as an environment

$$\begin{aligned} \mathsf{var} &\overset{\text{def}}{=} \{y, z\} \\ l &\overset{\text{def}}{=} \{(y, c), (z, a)\} \end{aligned} \tag{27.2}$$

The environment l is a mapping, so it can be invoked as a function. When we evaluate the l of Environment 27.2 on variable y, we get the value c, or

$$l(y) = c$$

It is easy to extend an environment in predicate logic. For example, in Environment 27.2, we do not have a variable x. If we needed this new variable to be in the environment, and we want it to map to the value c, we can add this information into the environment by the statement

$$l[x \mapsto c] \tag{27.3}$$

Combining Environment 27.2 with Extension 27.3, we get a new environment

$$\begin{aligned} \mathsf{var}' &\overset{\text{def}}{=} \{y, z, x\} \\ l' &\overset{\text{def}}{=} \{(y, c), (z, a), (x, c)\} \end{aligned} \tag{27.4}$$

We can see that the mappings l and l' each have a finite set as the domain and are "into" mappings, so the range is also finite. One useful way to represent a finite mapping – especially one that is defined extensionally as a set – is as a *look-up table*. We can now see that the examples of Environment 27.2 and Environment 27.4 are exactly look-up tables: given a variable in the domain of the mapping, we can look up the element from the universe of discourse A that is the model for the variable.

27.2 Logical Environments

This now allows us to define an environment and how to extend an environment to model a new variable.

Definition: Environment of Predicate Semantics

An *environment* for a universe A is a mapping $l : \mathsf{var} \to A$ from the set of variables var to values in A.

An *extended environment* of an environment l is a mapping $l[x \mapsto a]$ that maps a new variable x to $a \in A$ and that maps any distinct variable y to $l(y)$.

To see how to use an environment, consider the partial model of base-4 arithmetic that we introduced in a previous class. We had $A = \{0, 1, 2, 3\}$ and a predicate $\underline{P}^{\mathcal{M}}$ that was interpreted as the set $P^{\mathcal{M}}$ of even numbers in A.

Is there an environment in which $\underline{P}^{\mathcal{M}}(x)$ holds? That is, is there a mapping of x into $P^{\mathcal{M}}$?

Such an environment can be created from an existing environment, which could be the empty environment. We can use either of the semantic statements

$$l[x \mapsto 0]$$
$$l[x \mapsto 2]$$

to accomplish our goal. In the base-4 example, we also had a function $f^{\mathcal{M}}(\cdot)$ that added 1 to its input, modulo 4; the mapping $l(x)$ gives us the ability to reason about $P^{\mathcal{M}}(f^{\mathcal{M}}(x))$ and other, more elaborate, formulas.

We will use an abbreviation, not in the textbook, that is in common use in semantics.

Definition: Interpretation

An *interpretation* $\mathcal{I}$ is a model $\mathcal{M}$ combined with an environment l.

Example: An environment for a proof.

It is reasonable to ask why we might need environments, or interpretations, at all. Is it not sufficient to just provide a model?

Answers to this question vary from simple to complicated. The deep reason for needing an environment is to model computer programs, which may reference variables outside of the scope of a code fragment. An interpretation is a way to ensure that there is a legitimate implementation for the program.

Another reason to use an interpretation is so that we can formulate the semantics of a proof.

Consider the proof for a sequent that was proved earlier in the course:

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

This proof used two "fresh" variables, w and z. If we want a single model that captures the semantics of this proof, it must be able to describe the formula in each line of this proof. Some lines will contain a variable that is unbound in the corresponding line. At the very least, we will need to have a set that includes the symbols for the "fresh" variables, such as

$$\mathsf{var} \stackrel{\text{def}}{=} \{w, z\}$$

If one line of the proof depends on the use of the rule for existential elimination, then an interpretation will need a model $\mathcal{M}$ for the proof, and will also need to have an environment $\mathcal{I}$ for that variable. Suppose that, for a model $\mathcal{M}$, the universe of discourse $\mathcal{A}$ has as a as a member; the environment for $\mathcal{M}$ might include

$$l[z \mapsto a]$$

By carefully specifying the model $\mathcal{M}$, including the necessary variables and the environment, a human – or a computer! – could verify that each line of the proof evaluates to $\mathtt{T}$ in the interpretation $\mathcal{I}$.

Another kind of proof that needs an interpretation is a proof that uses a free variable. This type of proof is unusual but is also legitimate. Semantics that capture every line of the proof will need an environment that includes the free variable in the set var. By having a model and an environment, we can ensure that there is at least one way to interpret the proof.