

Structures: Still More Notes on Leary, Chapter 1

Curtis Brown

February 4, 2002

1 Defining Structures

Now the plot begins to thicken. The previous handout, “Defining a Language,” was exclusively concerned with the *syntax* of a language \mathcal{L} . That is, we defined a language by carefully delimiting which strings of basic symbols in \mathcal{L} are formulas (or what logicians sometimes call “well-formed formulas” or “wffs” for short). But we haven’t said anything yet about what these formulas mean. We need a way to specify not only the syntax of a language, but also its semantics; not only which strings are grammatical, but also what their content is. This is what *structures* are for.

Meaning or semantics has to do with the relation between language and the world, between language and what it represents. The only nonlogical parts of a first-order language, i.e. the only items that can vary from one first-order language to another, are the constant symbols, relation symbols, and function symbols. A semantics for such a language will need to specify what these symbols express. Intuitively, constants represent particular things; function symbols represent functions; and relation symbols represent relations. (Recall the set-theoretic analysis of relations, according to which an n -place relation is a set of ordered n -tuples, and an n -place function is an $n + 1$ -place relation that satisfies the condition that no two tuples differ only in their last place.)

So there shouldn’t be anything too surprising in the definition of a structure (Leary’s Definition 1.6.1, p. 26):

Definition: Fix a language \mathcal{L} . An \mathcal{L} -**structure** \mathfrak{A} is a nonempty set A , called the **universe of \mathfrak{A}** , together with:

1. For each constant symbol c , of \mathcal{L} , an element $c^{\mathfrak{A}}$ of A
2. For each n -ary function symbol f of \mathcal{L} , a function $f^{\mathfrak{A}} : A^n \rightarrow A$
3. For each n -ary relation symbol R of \mathcal{L} , an n -ary relation $R^{\mathfrak{A}}$ on A (i.e., a subset of A^n)

A couple of comments on the notation here may be in order. I don’t see that Leary explains the expressions A^n or $f^{\mathfrak{A}} : A^n \rightarrow A$. No doubt he assumes that his readers are

already familiar with these notational devices. A^n is simply the set of all the ordered n -tuples whose elements are elements of A . (Another way to put this involves the notion of the Cartesian product of sets. The Cartesian product $A \times B$ is the set of ordered pairs (x, y) where $x \in A$ and $y \in B$. So $A \times A$ is the set of ordered pairs both of whose members are elements of A . Similarly, $A \times A \times A$ is the set of ordered triples all of whose members are elements of A . Then A^n is the general case: A^n is $A \times A \times \dots \times A$ where A occurs n times.) To describe a function $f^{\mathfrak{A}}$ as $f^{\mathfrak{A}} : A^n \rightarrow A$ is to say that it is a function from A^n to A — that is, that the domain of the function is A^n and its range is A .

A simple example: suppose that A is $1, 2$. Then A^n is

$$\{(1, 1), (1, 2), (2, 1), (2, 2)\}$$

We can define a binary function \heartsuit that has A^n as its domain: that is, the possible pairs of arguments for the function $x\heartsuit y$ are $(1, 1), (1, 2), (2, 1)$, and $(2, 2)$. Similarly, the function has A as its range — that is, the possible values of the function (relative to the universe A) are 1 and 2 . (For example, \heartsuit might be the **max** function; in that case $x\heartsuit y$ takes the value 1 for the argument pair $(1, 1)$ and takes the value 2 for the other three argument pairs.)

2 Satisfaction

Things start to get a little ugly at this point, or at least a little complicated. But the notion of truth in a structure is extremely important. After all, we are ultimately interested in notions like validity and logical consequence, both of which have to do with truth. (For instance, a sentence C is a logical consequence of premises P_1, P_2, \dots, P_3 just in case C is true in *every* structure in which P_1, P_2, \dots, P_3 are all true.)

The basic idea of truth in a structure is clear enough. A structure is an interpretation or model of a language: it determines what individuals the constants refer to, and what functions and relations the function symbols and relation symbols express. (Keep in mind that functions and relations are just sets of n -tuples of individuals in the universe of the structure. So knowing what function or relation is expressed amounts to knowing what the value of the function is for every argument, and knowing which objects the relation relates. So, for instance, if we specify the relation that a two-place relation symbol *Larger* expresses, then this tells us, not that “*Larger*” has to do with size, but which objects in our universe are larger than which other objects.) Once we know what we are talking about, we should be able to determine whether what we are saying is true.

However, making the idea of truth in a structure clear and precise will require a blizzard of definitions. One thing that makes the matter more complex is that we will define truth not only for sentences, but also for formulas. Well, a formula is not true or false *simpliciter*. Consider (in the language of Tarski’s World) the formula *Larger*(x, y). Both variables in this formula are *free*, not bound by a quantifier. So it doesn’t make any sense to talk about the formula as being true or false: the variables do not refer to any object in particular (that’s why they are called “variables”), and are not bound by quantifiers, so the formula doesn’t say anything determinate (it does not “express a

complete thought,” as sentences are supposed to do). So how can we define truth in a structure for formulas? Only by relativizing it to an assignment of objects to variables. Suppose we have two objects: a , which is large, and b , which is small. Then we can say that ‘Larger(x, y)’ is true *relative to* the assignment $\{\langle x, a \rangle, \langle y, b \rangle\}$, but false relative to the assignments $\{\langle x, b \rangle, \langle y, a \rangle\}$, $\{\langle x, a \rangle, \langle y, a \rangle\}$, and $\{\langle x, b \rangle, \langle y, b \rangle\}$. (Usually we will express this by saying that the assignment just specified *satisfies* the formula.) This is the idea that the first few definitions will make precise.

So, here goes (Leary’s Definition 1.7.1, p. 32):

Definition: If \mathfrak{A} is an \mathcal{L} -structure, a **variable assignment function into \mathfrak{A}** is a function s that assigns to each variable an element of the universe A . So a variable assignment function into \mathfrak{A} is any function with domain $Vars$ and codomain A .

Leary notes that variable assignment functions need not be injective or bijective. These terms are also used in problem 5 on p. 31. This is another case in which Leary assumes background that we may or may not have. A function is *injective* if it does not give the same value for more than one argument. Recall that no function gives more than one value for a given argument. Putting these two things together, we see that an injective function must be *one-to-one*: for each argument, there is only one value, and for each value, there is only one argument. A function is said to be *surjective* if it is “onto,” that is, if every item in the range (or “codomain”) of the function is a value of the function for some argument. A function is said to be *bijective* if it is both injective and surjective, that is, both one-to-one and onto. So the point here is that variable assignment functions need not determine a different value for each argument, and also need not take every element of A as its value for some argument.

So far we’re just assigning an object to each distinct variable. We aren’t yet considering how we will use this assignment to discuss truth of a formula relative to an assignment. Next we introduce a notation for taking a variable assignment function s and changing it for a single variable. (This is Leary’s Definition 1.7.2, p. 33.)

Definition: If s is a variable assignment function into \mathfrak{A} and x is a variable and $a \in A$, then $s[x|a]$, called an **x -modification of the assignment function s** , is the variable assignment function into \mathfrak{A} defined as follows:

$$s[x|v](v) = \begin{cases} s(v) & \text{if } v \neq x \\ a & \text{if } v = x \end{cases}$$

Consider the following assignment function s (mentioned earlier): $\{\langle x, a \rangle, \langle y, b \rangle\}$. Now suppose we want to modify this function so that both variables are assigned a . We can write this modified variable assignment function like this: $s[x|b]$. Then $s(x) = a$, but $s[x|b](x) = b$. On the other hand, $s(y) = b$ and $s[x|b](y) = b$. Since the modification affects only the variable x , applying the modified variable assignment function to y gives us the same results as applying the original variable assignment function to y .

We are gradually closing in on a definition of truth of a formula relative to an assignment. If we have a formula with some free variables, then assigning an element from the domain to each free variable will suffice to determine the interpretation of all the terms. There are three kinds of terms: constants, variables, and function symbols followed by terms. Constants already have an interpretation assigned by the structure itself, so they are taken care of. The variables are taken care of by the variable assignment function s . This leaves only the complex terms constructed out of function symbols and other terms. But once the interpretation of constants and variables is taken care of, the complex terms are taken care of also. Each term t_i that follows a function symbol is a constant, variable, or complex term. If it is a variable or a constant, it is taken care of already. If it is a complex term, we look at *its* constituent terms. Recursively following this procedure, we eventually reach function symbols all of whose terms have interpretations. Then we read the interpretation of the function symbol followed by those terms off the structure. Then we return to the next higher level of our recursive process and proceed to examine the next term. Eventually we will have an interpretation for every term in our formula. [Note to self: clarify the description of this recursive procedure! Leary notes that there is actually a theorem, the aptly named “Recursion Theorem,” that establishes that an assignment of referents to variables suffices to determine the interpretation of every term, and hints that it’s a little tricky. But it should be possible to describe the recursive process in a clear and detailed enough way that it is intuitively clear that we are guaranteed to be able to find an interpretation for every term.]

So we can take a variable assignment function, together with facts determined by the structure itself, and construct a more general term assignment function, as follows (Leary’s Definition 1.7.3).

Definition: Suppose that \mathfrak{A} is an \mathcal{L} -structure and s is a variable assignment function into \mathfrak{A} . The function \bar{s} , called the **term assignment function generated by s** , is the function with domain consisting of the set of \mathcal{L} -terms and codomain A defined recursively as follows:

1. If t is a variable, $\bar{s}(t) = s(t)$.
2. If t is a constant symbol c , then $\bar{s}(t) = c^{\mathfrak{A}}$.
3. If t is $ft_1t_2\dots t_n$, then $\bar{s}(t) = f^{\mathfrak{A}}(\bar{s}(t_1), \bar{s}(t_2), \dots, \bar{s}(t_n))$.

Now we can (finally) define what it is for a formula to be true relative to a variable assignment function. The usual term for truth relative to an assignment is *satisfaction*: we say that an assignment s satisfies a formula, or, in the language of the next definition, that a structure satisfies a formula *with* a particular assignment.

Definition: Suppose that \mathfrak{A} is an \mathcal{L} -structure, ϕ is an \mathcal{L} -formula, and $s : Vars \rightarrow A$ is an assignment function. Then \mathfrak{A} **satisfies ϕ with assignment s** , and write $\mathfrak{A} \models \phi[s]$, in the following circumstances:

1. If ϕ is $= t_1 t_2$ and $\bar{s}(t_1)$ is the same element of the universe A as $\bar{s}(t_2)$, or
2. If ϕ is $Rt_1 t_2 \dots t_n$ and $\langle \bar{s}(t_1), \bar{s}(t_2), \dots, \bar{s}(t_n) \rangle \in R^{\mathfrak{A}}$, or
3. If ϕ is $(\neg\alpha)$ and $\mathfrak{A} \not\models \alpha[s]$, (where $\not\models$ means “does not satisfy”), or
4. If ϕ is $(\alpha \vee \beta)$ and $\mathfrak{A} \models \alpha[s]$, or $\mathfrak{A} \models \beta[s]$ (or both), or
5. If ϕ is $(\forall x)(\alpha)$ and, for each element a of A , $\mathfrak{A} \models \alpha[s(x|a)]$.

If Γ is a set of \mathcal{L} -formulas, we say that \mathfrak{A} satisfies Γ with assignment s , and write $\mathfrak{A} \models \Gamma[s]$ if for every $\gamma \in \Gamma$, $\mathfrak{A} \models \gamma[s]$.

This definition is where we attach meanings to the logical symbols in \mathcal{L} . The structure gives the meaning of the constant symbols, function symbols, and relation symbols, while the definition of satisfaction above essentially gives the meaning of the symbols that are common to all first-order languages. These don't need to be given in the structure for the same reason that the logical symbols don't need to be listed when we specify a language: since they are common to all first-order languages, we can leave them out of the structures, which merely give the semantics of the symbols that can change from language to language.

Notice that the notion of an x -modification of a variable assignment function comes in handy in the final clause above. If we have a quantified sentence, we don't care what entity s assigns to occurrences of the variable that are bound by its quantifier. For example, if we have the formula

$$(\forall x)\text{Larger}(x, y)$$

and an assignment function

$$s = \{\langle x, a \rangle, \langle y, b \rangle\}$$

and we want to know whether the assignment function satisfies the formula, then we need to appeal to the function s to find how to interpret y , which is free, but the fact that s assigns a to x is irrelevant, because the quantifier tells us that the whole formula is true only if the subformula after the quantifier is true for every assignment of an object to x . The notation $s[x|a]$ gives us a convenient way to consider assignments that keep all of the assignments of s except its assignment to x . (In clause 5 above, ' $s[x|a]$ ' is written ' $s(x|a)$ ', i.e. with parentheses instead of square brackets, for readability: ' $\alpha[s(x|a)]$ ' is easier to read than ' $\alpha[s[x|a]]$ '!)

3 Models, Truth, Logical Implication, and Validity

We are almost ready, finally, to say what it means for a sentence to be true in a structure. First, we define what it is for a structure to be a *model* of a formula.

Definition: If ϕ is a formula in the language \mathcal{L} and \mathfrak{A} is an \mathcal{L} -structure, we say that \mathfrak{A} is a **model** of ϕ , and write $\mathfrak{A} \models \phi$, if and only if $\mathfrak{A} \models \phi[s]$ for every assignment function s . If Φ is a set of \mathcal{L} -formulas, we will say that \mathfrak{A} models Φ , and write $\mathfrak{A} \models \Phi$, if and only if $\mathfrak{A} \models \phi$ for each ϕ in Φ .

The symbol ‘ \models ’ is rather versatile! The right-hand side can be either a formula or a set of formulas. Much more versatility is yet to come.

Informally, the basic idea is that a structure is a model of a formula if the formula *must* be true in the structure, i.e. if the formula comes out true no matter what terms you substitute for its free variables.

Of course, sentences are special cases of formulas. In the case of sentences, it makes no difference what assignment function s we use. Why? Because sentences have no free variables, and assignment functions have no effect on bound variables. So if there is any assignment function s for which $\mathfrak{A} \models \phi[s]$ in the special case in which ϕ is a sentence, then, since it makes no difference what assignment function we use, ϕ will be true for every assignment function, and hence we can simply say that $\mathfrak{A} \models \phi$, period.

To make it obvious when we are dealing with formulas that are also sentences, we will use a different Greek variable for sentences, ‘ σ ’. (Notice the alliterative choice of Greek letters: phi, with its initial ‘f’ sound, for formulas, and sigma, with its initial ‘s’ sound, for sentences.)

Definition: If σ is a sentence, then we say that σ is **true** in \mathfrak{A} if and only if $\mathfrak{A} \models \sigma$, which in turn is the case if and only if there is any assignment function s for which $\mathfrak{A} \models \sigma[s]$.

Notice that a structure models a formula if and only if the sentence that results from prefixing to the formula universal quantifiers binding all the free variables of the formula results in a sentence that is true in the structure.

Finally, we turn to Leary’s definitions of logical implication and validity. Logical implication is defined in Definition 1.9.1 on p. 43:

Definition: Suppose that Δ and Γ are sets of \mathcal{L} -formulas. We will say that Δ **logically implies** Γ and write $\Delta \models \Gamma$ if for every \mathcal{L} -structure \mathfrak{A} , if $\mathfrak{A} \models \Delta$, then $\mathfrak{A} \models \Gamma$.

If we restrict ourselves to sentences for a moment, we can say that one set of sentences Δ logically implies another set of sentences Γ if and only if every structure in which all the sentences Δ are true is a structure in which all the sentences in Γ are true. Equivalently, we can say that Δ logically implies Γ if and only if every model of Δ is also a model of Γ .

Still another way to say the same thing: sometimes structures are called *interpretations*, and a model of a sentence is called an interpretation in which the sentence is true. So we can say that Δ logically implies Γ iff every interpretation that makes all the sentences in Δ true also makes all the sentences in Γ true.

It is a short step to the notion of a *valid argument*: an argument from a set of premises

Γ to a conclusion C is valid iff $\Gamma \models C$.

The term ‘valid’ is also used for a property of individual formulas. This usage is defined in Leary’s Definition 1.9.2:

Definition: An \mathcal{L} -formula ϕ is said to be **valid** if $\emptyset \models \phi$, in other words, if ϕ is true in every \mathcal{L} -structure with every assignment function s . In this case, we will write $\models \phi$.

Notice the relation between this definition and Barwise and Etchemendy’s more informal definition in *Language, Proof, and Logic*. B & E informally characterize a valid sentence as one that remains true regardless of which predicates are used. The idea of substitution of predicates is an informal way to accomplish what the formal definition accomplishes by considering different structures (which will have different interpretations of the predicates).