

Defining a Language: Notes on Leary, Chapter 1

Curtis Brown

February 4, 2002

1 Basic Symbols

Leary's Definition 1.2.1 defines a first-order language as a collection of symbols. This is maybe a little misleading, as the later definitions of terms (Def. 1.3.1), formulas (Def. 1.3.3), and sentences (Def. 1.5.3) are also needed to fully specify a language.

First-order languages all share certain features: they share many of their basic symbols, and they share a syntax (that is, they share the definitions of terms, formulas, and sentences in terms of the basic symbols and how they are combined). However, they also differ in a few respects. The general definition of a language does not specify what particular constant symbols, relation symbols, and function symbols are to be included. Since all the *other* features of a language are shared among all languages, we don't need to mention them every time we specify a language; we only need to specify the features that are specific to this particular language. Therefore, we can specify a language by specifying a set of constant symbols, relation symbols, and function symbols.

(If you have studied Barwise and Etchemendy, *Language, Proof, and Logic*, notice that Leary's definition of a language is very similar to Barwise and Etchemendy's. They mention in Chapter 1 that it is more accurate to speak of *languages* (plural) of first-order logic rather than of a *language* (singular) of first-order logic, and they introduce several specific languages: the blocks language, with predicates like $Tet(x)$ and $Larger(x,y)$; a language to discuss their family members and pets, with individual constants like 'max' and 'claire', and predicates like 'Gave(x,y,z,t)'; the language of set theory, with constants for the natural numbers and a predicate for set membership; and so on. The main *differences* between B&E's definition and Leary's are: (1) Leary's presentation is a bit more formal; (2) Leary's first-order languages include function symbols, whereas B&E leave these out of their official language, although they mention them from time to time; (3) Leary's languages are a little more stripped-down, for example including the logical connectives \vee and \neg , but not including \wedge , \rightarrow , or \leftrightarrow . This is to keep the official definition of a language as simple as possible, to make proofs easier. We can treat the things that have been left out as simply abbreviations for expressions using the symbols we do have; for example, we can treat $P \wedge Q$ as simply an abbreviation for $\neg(\neg P \vee \neg Q)$. Similarly, our official language contains only one quantifier, \forall , but we can regard the quantifier \exists as simply an abbreviation: $(\forall x)$ abbreviates $\neg(\exists x)\neg$.)

For convenience, here is the list of basic symbols that constitute the building blocks of a first-order language (this is Leary's Definition 1.2.1).

Definition: A **first-order language** \mathcal{L} is an infinite collection of distinct symbols, no one of which is properly contained in another, separated into the following categories:

1. *Parentheses:* $(,)$.
2. *Connectives:* \vee, \neg .
3. *Quantifier:* \forall .
4. *Variables, one for each positive integer n :* $v_1, v_2, \dots, v_n, \dots$. The set of variable symbols will be denoted $Vars$.
5. *Equality symbol:* $=$.
6. *Constant symbols:* Some set of zero or more symbols.
7. *Function symbols:* For each positive integer n , some set of zero or more n -ary function symbols.
8. *Relation symbols:* For each positive integer n , some set of zero or more n -ary relation symbols.

There are a couple of things to notice about this definition. A fussier version of this definition would indicate that the symbols are being mentioned rather than used by putting single quotation marks around them, e.g. *Parentheses:* $'(,)'$. I have already mentioned that missing symbols such as $'\wedge'$ can be defined in terms of the symbols provided here. Notice that all the basic symbols are specified except the constant symbols, function symbols, and relation symbols. (Actually, of course, the "equality symbol" *is* a relation symbol, but we list it separately because every first-order language must contain it.) "Equality symbol" may not be the best term for $'='$ – I would prefer "identity symbol" to emphasize that a sentence of the form $'v_1 = v_2'$ is true if and only if the item denoted by the left-hand side is *the very same thing* as the item denoted by the right-hand side. In the last two items on the list, notice that the phrase "for each positive integer n " merely indicates that we will have zero or more functions and relations of each possible arity (i.e. one-place or unary functions and relations; two-place or binary functions and relations; and so on). The phrase "zero or more" of course guarantees that we don't need to provide any of these additional symbols if we don't need them!

Since everything except the constant, function, and relation symbols is prespecified, we can specify a language by merely stating which constant, function, and relation symbols we will add to the materials already provided. Note that we collect these symbols into a set in specifying a language; thus Leary notes that the language of set theory is just $\{\in\}$. (Actually, this isn't quite correct; as Barwise and Etchemendy note in chapter 15, we really need two additions to the basic materials of a first order language. It would be better to describe this language as the language $\{\in, Set()\}$. Barwise and Etchemendy's

two styles of variables are a kind of abbreviation: using a variable a, b, \dots amounts to using an ordinary variable x and stating that $Set(x)$.)

Leary notes that in the general case, the specification of a language will be

$$\{c_1, c_2, \dots, f_1^{a(f_1)}, f_2^{a(f_2)}, \dots, R_1^{a(R_1)}, R_2^{a(R_2)}, \dots\}$$

This looks pretty weird at first, but it's really not so bad. Of course, c_1 is the first constant, c_2 is the second constant, and so on. Similarly, $f_1^{a(f_1)}$ is the first function symbol, and the superscript $a(f_1)$ is just a way of indicating the arity of the function.

We could almost save ourselves some symbols and put it this way instead: the specification of a language is

$$\{c_1, c_2, \dots, f_1^1, f_2^1, \dots, f_1^2, f_2^2, \dots, \dots, R_1^1, R_2^1, \dots, R_1^2, R_2^2, \dots, \dots\}$$

where the superscript indicates the arity of the function or relation and the subscript indicates whether it is the first, second, etc. function of that arity. The problem is that perhaps this doesn't make it clear enough that we are not stopping with functions and relations of arity two. (I've tried to remedy this with double sets of ellipses, but that may be hard to interpret!)

2 Terms

Now that we have the basic building blocks of a language, we need to specify the syntax of the language. That is, we need to say how we can use the building blocks to form more complex expressions. Our definitions will be recursive, which will (a) insure that we could construct an infinite number of distinct expressions even if we began with finite building blocks, and (b) enable us to prove things about expressions in a language by using mathematical induction. More on this a little later in the course.

So here is the definition of a term (Leary's Definition 1.3.1, p. 11):

Definition: If \mathcal{L} is a language, a **term** of \mathcal{L} is a nonempty finite string t of symbols from \mathcal{L} such that either:

1. t is a variable, or
2. t is a constant symbol, or
3. t is $ft_1t_2\dots t_n$, where f is an n -ary function symbol of \mathcal{L} and each of the t_i is a term of \mathcal{L} .

This is pretty straightforward. We have two base cases of terms, namely variables and constant symbols. And then we can recursively build more terms by using function symbols: an n -ary function symbol whose argument places are filled by n terms (not necessarily distinct) will also be a term.

Notice that in the official language, we don't use parentheses and commas; we construct a term simply by stringing the appropriate number of terms after a function symbol.

However, for perspicuousness, we will usually use parentheses and commas in the familiar way. Thus $f(t_1, t_2)$ will be regarded as just a longer version of the canonical form ft_1t_2 . Similarly, $f_1(t_1, f_2(t_2, t_3))$ will be regarded as a longer version of the canonical $f_1t_1f_2t_2t_3$. The parentheses and commas make things easier to read, but they aren't necessary since the official version is also unambiguous.

3 Formulas

Here is Leary's Definition 1.3.3 (p. 13) for formulas. (Note: we will later distinguish between two kinds of formulas, sentences and non-sentences. Both are sentence-like; the only difference is that the non-sentences contain free variables, i.e. variables unbound by a quantifier. Thus they don't "express a complete thought," to revert to the definition of a sentence I learned in middle school English.

Definition: If \mathcal{L} is a first-order language, a **formula** of \mathcal{L} is a nonempty finite string ϕ of symbols from \mathcal{L} such that either:

1. ϕ is $=t_1t_2$, where t_1 and t_2 are terms of \mathcal{L} , or
2. ϕ is $Rt_1t_2\dots t_n$, where R is an n -ary relation symbol of \mathcal{L} and t_1, t_2, \dots, t_n are all terms of \mathcal{L} , or
3. ϕ is $(\neg\alpha)$, where α is a formula of \mathcal{L} , or
4. ϕ is $(\alpha \vee \beta)$, where α and β are formulas of \mathcal{L} , or
5. ϕ is $(\forall v)(\alpha)$, where v is a variable and α is a formula of \mathcal{L} .

There are a few minor points to make here. The first clause states that $=t_2t_2$ is a term. This looks weird, but it is just an instance of the official syntax of our language. '=' is a special instance of a two-place relation symbol, and like the other binary relation symbols (as treated in the next clause), it yields a formula when it is followed by two terms. Of course, we would ordinarily write this using infix notation rather than prefix notation, as ' $t_1 = t_2$ '.

Another point to notice is that the official language requires parentheses around every compound formula. (The atomic formulas are those defined by clauses 1 and 2; the rest are nonatomic or compound. Any formula constructed out of other formulas together with a connective or a quantifier is compound.) This is different than the convention in Barwise and Etchemendy, but it makes things simpler. Informally, we can drop some of these parentheses, e.g. the outermost parentheses, as long as this does not hurt readability or introduce ambiguity.

4 Sentences and Non-Sentences

The previous handout gave the definitions of a language, of terms, and of formulas. Now we need to distinguish between two kinds of formulas, sentences and non-sentences. (The non-sentences are sometimes called "open sentences.")

We begin by defining a *free variable*. The following is (almost) Leary's Definition 1.5.2, p. 23.

Definition: Suppose that v is a variable and ϕ is a formula. We will say that v is **free in ϕ** if and only if:

1. ϕ is atomic and v occurs in (is a symbol in) ϕ , or
2. ϕ is $(\neg\alpha)$ and dv is free in α , or
3. ϕ is $(\alpha \vee \beta)$ and v is free in at least one of α or β , or
4. ϕ is $(\forall u)(\alpha)$ and v is not u and v is free in α .

Leary's definition has "if" where the one above has "if and only if," but the latter seems clearly to be his intention. (Mathematicians seem to sometimes use simply "if" where logicians with a background in philosophy would write "if and only if" or its abbreviation "iff".)

The basic idea here is that an occurrence of a variable v is free if it is not bound by a quantifier, i.e. if it is not within the scope of a quantifier $(\forall v)$ (or a quantifier $(\exists v)$, but that is not part of our official language \mathcal{L}). (Of course, being within the scope of a different quantifier $(\forall u)$, where $u \neq v$, does not prevent v from being free.) Another way to say this is that v is free if it is not *bound* by a quantifier.

The text mostly discusses "free variables" rather than free *occurrences* of a variable, but the latter term is a little more accurate, since the same variable might be free in one occurrence and bound in another (as exercise 6 on p. 25 makes clear).

Now we can define *sentence* very simply:

Definition: A **sentence** in a language \mathcal{L} is a formula of \mathcal{L} that contains no free variables.

A bit more precisely, a sentence is a formula that contains no free occurrences of any variable.