

1. *First-order logic.* This subject is also called *elementary* logic. I use these terms interchangeably. Most of its principles were developed during the late 1800s, but crystallized into a working framework only in the late 1920s. By the 1940s it had become the default system. It was intended to mirror practice in expounding many algebraic theories, where the use of set theory could be minimized. It is convenient, intuitive, simple enough to be explained briefly, and its default use highlights as exceptional some mathematical theories for which set theory is actually necessary.
2. *Syntax of elementary languages.* Rather than adapting one vocabulary for elementary logic, I provide many options for the construction of elementary languages for conveying algebraic theories. I continue to use, as appropriate, some of the syntactical notions introduced earlier, in the discussion of Boolean logic. But I *don't* use Boolean variables.
 - a. The alphabet of an elementary language for number theory might include
 - i. symbols to represent variables that range over the natural numbers,
 - ii. a symbol that with a variable will represent universal quantification,
 - iii. symbols representing natural numbers zero, one, and two, that will be used as constants,
 - iv. a symbol representing the operation that transforms each natural number to the next greater one,
 - v. symbols representing the binary addition and multiplication operations on natural numbers,
 - vi. symbols representing the binary equality and order relations on natural numbers,
 - vii. a symbol representing negation, and
 - viii. a symbol representing implication.
 - b. I'll denote by $=$ and $\forall$ the strings whose single members are occurrences of the equality and universal quantification symbols. Strings whose single members are occurrences of constant symbols are called *constants*. For each positive integer n , strings whose single members are occurrences of n -ary relation or operation symbols are called *n -ary predicates* or *operators*, respectively.
 - c. As you know, the other Boolean connectives can be defined in terms of those mentioned, so no further symbols for them are required. Moreover, the existential quantifier can be defined from the universal and negation, so no symbol for it is required: an expression of the form $\exists x p$ will be regarded as an abbreviation for $\neg \forall x \neg p$.
 - d. This example elementary language is imitated to devise languages for many subjects.
 - e. Some authors give the equality symbol no special status. Following Tarski's practice, however, I will regard it as a special one, included along with the

- quantification, implication, and negation symbols in the alphabet of *every* elementary language.
- f. Some elementary languages may employ no other relation symbols. Some may use ternary relation symbols—for example, to indicate when three points should be collinear. Some may use quaternary relation symbols—for example, to indicate when the segment between two points is congruent to that between two others. I suspect that elementary languages for software applications may commonly use relation symbols with finite arity greater than four.
 - g. Some elementary languages may employ no operation symbols. Some may use ternary operation symbols—for example, to represent the fourth harmonic of three points on a line. I suspect that elementary languages for software applications may commonly use operators with finite arity greater than three.
 - h. Some elementary languages may employ no constant symbols. Some authors use symbols for operations with arity zero, and regard those as constants.
 - i. Some authors use binary and ternary relation symbols, for example, to represent singular and binary operations. For example, $Sabc$ might represent the statement $a = b + c$.
 - j. The alphabet of an elementary language *must* include symbols for infinitely many variables. While elementary syntax can be discussed with only finitely many, the corresponding semantics is sufficiently different to warrant calling that a separate area of logic.
 - k. These, then, are the only restrictions on an elementary language:
 - i. infinitely many symbols for variables,
 - ii. individual symbols for universal quantification, equality, negation, and implication, and
 - iii. disjointness of the specified sets of various kinds of symbols.
 - l. For an extremely simple example, see the unit *Zermelo–Fraenkel Set Theory*: the set theory we are using can be expressed with an elementary language that has
 - i. a countable number of variables,
 - ii. no constant or operation symbols,
 - iii. only one relation symbol besides that for equality, namely the symbol for the binary membership relation.
 - m. As a second example the alphabet of an elementary language for linear algebra over a field of scalars might include
 - i. a constant symbol to represent the zero vector,
 - ii. singular operation symbols representing multiplication by each scalar,
 - iii. and a binary operation symbol representing vector addition.

For real or complex linear algebra, that language would employ 2^{ω} singular operation symbols. With it, for example, an expression of the form $x - y$ would be regarded as denoting the string $+x \cdot_{-1} y$. Here, prefix notation is in use, x and y are variables, and $+$, $\cdot_{-1}$ are the strings whose single members are occurrences of the vector addition symbol and the symbol for multiplication by the scalar -1 .

3. *Terms*

- a. The set of *terms* of an elementary language is defined recursively as follows.
 - i. All variables and constants are terms.
 - ii. For each positive integer n , if o is an n -ary operator and $t_1, \dots, t_n$ are terms, then the string $ot_1 \dots t_n$ formed by concatenating o and $t_1, \dots, t_n$ is a term.
- b. A term is called *constant* if no variable symbol occurs in it.
- c. *Theorem.* The cardinal of the set of all terms is the sum σ of the cardinals of the sets of all variables, constants, and operators.
 - i. *Proof.* For each positive integer n , let T_n denote the set of all terms with fewer than n occurrences of operation symbols, and O_n be the set of all n -ary operators. I shall prove $\#T_n \leq \sigma$ by recursion on n .
 - (1) First, $\#T_1 \leq \sigma$ because T_1 is merely the union of the sets of variables and constants. This also confirms that σ is infinite.
 - (2) Now suppose $\#T_m \leq \sigma$ for each $m < n$. Each term in $T_n - T_{n-1}$ is a string $ot_1 \dots t_m$ for some unique $m < n$ and $o \in O_m$, and unique terms $t_1, \dots, t_m \in T_{n-1}$. Thus,

$$\begin{aligned} \#T_n &= \#T_{n-1} + \#(T_n - T_{n-1}) \leq \sigma + \sum_{m < n} (\#O_m) \sigma^m \\ &\leq \sigma + \sum_{m < n} \sigma^{m+1} \leq \sigma + (n-1)\sigma^n = \sigma. \end{aligned}$$

It follows that the cardinal of the set of all terms is

$$\#\bigcup_n T_n \leq \sum_n \#T_n \leq \sum_n \sigma = \omega\sigma = \sigma.$$

Now select a variable v and let S be the set of all terms consisting of an n -ary operator o concatenated with n copies of v . Since the sets V of all variables, C of all constants, and S are disjoint,

$$\#\bigcup_n T_n \geq \#V + \#C + \#S = \#V + \#C + \#\bigcup_n O_n = \sigma.$$