1 Introduction

Logic surely has something to do with reasoning. We say that someone has reasoned poorly about something if they have not reasoned logically, or that an argument is bad because it is not logically valid. There has been much speculation over just what types of logical systems are appropriate for guiding deductive reasoning. Some have suggested that classical first-order logic is the ideal for guiding reasoning (for example, see Quine (1986), Resnik (1996) or Rumfitt (2015)).[1] Classical first-order logic has occasionally been dubbed "the one true logic."

Though there is much debate about the underlying issues, it is safe to say that classical first-order logic has been prevalent in mathematics and philosophy over the past century or so. There are good reasons for this. Classical first-order logic has rules which are more or less intuitive, and it is surprisingly simple given its strength. Plus, it is both sound and complete.[2] As is common in philosophy, however, there is no consensus on the primary status of classical first-order logic. As indicated in Section 6, there are certain expressive limitations to first-order logic and, as indicated in the later sections (7-9), there are serious, well-developed rivals to classical logic.

This Element will examine classical first-order logic and then provide a more thorough understanding of it by exploring some alternative logics. In the first half, we provide the details of classical first-order logic. Then, we consider three alternatives to the system we develop: classical higher-order logic (Section 7), intuitionistic logic (Section 8), and para-consistent logics (Section 9).[3]

CLASSICAL FIRST-ORDER LOGIC

2 Formal System

For our purposes, a logic includes a formal language and a deductive system and/or a model-theoretic semantics.[4] A formal language is like a natural

[1] When discussing critical thinking and argument analysis, classical first-order logic, is usually, but not always, the starting point. See, for example, Stebbing (1939) and Woods (2021).

[2] We say what this means below.

[3] There are a number of alternatives to classical first-order logic that we will not have space to cover here. In particular, we omit modal logics (see Marcus (1995), for example) and substructural logics (see Restall (2000), amongst others).

[4] Sections 2-6 are based on Shapiro and Kouri Kissel (2020). More complete proofs and proof sketches can be found there, or in most textbooks for formal logic. We include here only a few of the complete proofs: those that are required to prove unique readability, as an example of how proofs often go, and a proof of theorem 4.5 (The rule of Cut) Not all logics have both a deductive system and model theory, but we will develop both for classical first-order logic here.

language, such as English, Japanese, or Swahili, but with explicit and simple rules of composition. It has terms and components which come together in a grammatical way. The deductive system is meant to capture reasoning presented in the language. The model theory gives meanings to the terms in the language, and tells us which propositions are true in various interpretations.

Section 3 develops a formal language with a rigorous syntax and grammar. The formal language is a recursively defined collection of strings on a fixed alphabet. Items in the formal language do not mean anything on their own – they need the deductive system and model theory to give them meaning. Some items correspond to items in natural language, and the deductive system and model theory are designed to preserve this correspondence, at least approximately. An *argument* is a non-empty collection of sentences in the formal language, one of which is designated to be the *conclusion*. The other sentences (if any) in an argument are its *premises*.

Section 4 sets up a deductive system for the language, in the spirit of natural deduction. An argument is *derivable* if there is a deduction from some or all of its premises to its conclusion. Section 5 provides a model-theoretic semantics. An argument is *valid* if there is no interpretation (in the semantics) in which its premises are all true and its conclusion false. This reflects the long-standing view that a valid argument is truth-preserving.

In Section 6, we explore the relationship between the deductive system and the model-theoretic semantics. We will be interested mostly in the relationship between derivability and validity. We sketch two important theorems. The first, *soundness*, will show that no deduction can start with a true premise and end with a false conclusion. So, if an argument is derivable, then it is valid. Proving this shows that deductions preserve truth. The second, *completeness*, is the converse of soundness. It shows that if an argument is valid, then it is derivable. This demonstrates that there are “enough deductions”: all valid arguments are derivable. We also briefly indicate other features of the logic, some of which are corollaries to soundness and completeness.

3 Language

Here we develop the basics of the formal language we will use for classical logic. As noted, a formal language is a recursively defined set of strings on a fixed alphabet. Some parts of the formal language correspond to some parts of natural language. This correspondence is not really a part of the formal language, but noting these correspondences can help motivate the system.

3.1 Building Blocks

We start with analogues of *singular terms*, sometimes just called *terms*. These are linguistic items whose function is to denote a person or object. We assume a stock of *individual constants*. These are lower-case letters, near the beginning of the Roman alphabet, with or without numerical subscripts:

$a, a_1, b_{23}, c, d_{22}$, etc.

Constants are an analogue of proper names. We allow ourselves an infinity of individual constants. Each constant is a single character, and so individual constants do not have an internal syntax. Thus, we have an infinite alphabet.[5]

There are two roles that constants play. One is like that of proper names, where a constant denotes a specific object or person (in each interpretation). The other is to denote specific, but unspecified (or arbitrary), objects and persons.[6]

We also assume a stock of *individual variables*. These are lower-case letters, near the end of the alphabet, with or without numerical subscripts:

w, x, y_{12}, z, z_4, etc.

Variables are used to express generality. In ordinary language, some uses of pronouns play the latter role, as in "When a dog is angry, it growls."[7]

We next introduce *function symbols*. These allow complex terms corresponding to: "5 + 3" and "the Marvel character played by Chadwick Boseman," or terms containing variables, like "the sister of x" and "x/y." Function symbols are lower-case letters, near the middle of the alphabet:

f, g, h, etc.

Each function has an arity, that is, the number of arguments it takes.

We now give a recursive definition of the *terms* of the language:

1. All individual constants and all variables are terms.
2. If $t_1, \ldots t_n$ are terms and f is an n-place function symbol, then $ft_1 \ldots t_n$ is a term.
3. That's all folks: every term is constructed in accordance with (1) and (2).

A term is *closed* if it contains no variables.

For each natural number n, we introduce a stock of $n-$place *predicate letters*. These are upper-case letters at the beginning or middle of the alphabet.

5 This could be avoided by taking a constant like d_{22}, for example, to consist of three characters, a lowercase "d" followed by a pair of subscript "2"s.

6 Some authors use (free) variables for this role; others use different symbols for this, sometimes called "individual parameters."

7 Another use of pronouns is to denote a specific object or person, as supplied by the context. The formal language has no analogue of this.

A superscript indicates the number of places, and there may or may not be a subscript. For example, A^3, B^3_2, D^3, etc, are three-place predicate letters. We often omit the superscript when no confusion will result. We also add a special two-place predicate symbol "=" for identity.

Zero-place predicate letters are sometimes called "sentence letters." They correspond to freestanding sentences whose internal structure does not matter. One-place predicate letters, called "monadic predicate letters," correspond to linguistic items denoting properties, adjectives, or common nouns like "woman," "large," or "prime number." Two-place predicate letters, called "binary predicate letters," correspond to linguistic items denoting binary relations, like "is a parent of" or "is shinier than." Three-place predicate letters correspond to three-place relations, like "lies on a straight line between." And so on.

The *non-logical terminology* of the language consists of its individual constants and predicate letters. The symbol "=", for identity, is not a non-logical symbol. In taking identity to be logical, we provide explicit treatment for it in the deductive system and in the model-theoretic semantics. Most authors do the same, but there is some controversy over the issue (see Quine (1986) Chapter 5). If K is a set of constants and predicate letters, then we give the fundamentals of a language $\mathcal{L}1K\!=$ built on this set of non-logical terminology. It may be called the *first-order language with identity* on K. A similar language that lacks the symbol for identity (or which takes identity to be non-logical) may be called $\mathcal{L}1K$, the *first-order language without identity* on K.

3.2 Atomic Formulas

If V is an n-place predicate letter in K, and $t_1, \ldots, t_n$ are terms of K, then $Vt_1 \ldots t_n$ is an *atomic formula* of $\mathcal{L}1K\!=$. Examples of atomic formulas include:

$P^4xaab, C^1x, C^1a, D^0, A^3aba$.

The last one is an analogue of a statement that a certain relation (A) holds between the objects $\langle a, b, a\rangle$. If t_1 and t_2 are terms, then $t_1 = t_2$ is also an atomic formula of $\mathcal{L}1K\!=$. It corresponds to an assertion that t_1 is identical to t_2.

If an atomic formula has no variables, then it is called an *atomic sentence*. If it does have variables, it is called *open*. In the above list of examples, the first and second are open; the rest are atomic sentences.

3.3 Compound Formulas

The final items of the lexicon are:

$\neg, \wedge, \vee, \rightarrow, \forall, \exists, (,)$

We give a recursive definition of a *formula* of $\mathcal{L}1K\!=$:

1. All atomic formulas of $\mathcal{L}1K{=}$ are formulas of $\mathcal{L}1K{=}$.
2. If θ is a formula of $\mathcal{L}1K{=}$, then so is $\neg\theta$.

A formula corresponding to $\neg\theta$ roughly "says" that it is not the case that θ. The symbol "¬" is called "negation," and is a unary connective.

3. If θ and ψ are formulas of $\mathcal{L}1K{=}$, then so is $(\theta \wedge \psi)$.

The wedge "∧" corresponds to the English "and" (when "and" is used to connect sentences). So $(\theta \wedge \psi)$ can be read "θ and ψ." The formula $(\theta \wedge \psi)$ is called the "conjunction" of θ and ψ.

4. If θ and ψ are formulas of $\mathcal{L}1K{=}$, then so is $(\theta \vee \psi)$.

The symbol "∨" corresponds to "either … or … or both," so $(\theta \vee \psi)$ can be read "θ or ψ." The formula $(\theta \vee \psi)$ is called the "disjunction" of θ and ψ.

5. If θ and ψ are formulas of $\mathcal{L}1K{=}$, then so is $(\theta \rightarrow \psi)$.

The arrow "→" roughly corresponds to "if … then … ," so $(\theta \rightarrow \psi)$ can be read "if θ then ψ" or "θ only if ψ."

The symbols "∧," "∨," and "→" are called "binary connectives," since they serve to "connect" two formulas into one. Some authors introduce $(\theta \leftrightarrow \psi)$ as an abbreviation of $((\theta \rightarrow \psi) \wedge (\psi \rightarrow \theta))$. The symbol "↔" is an analogue of the locution "if and only if."

6. If θ is a formula of $\mathcal{L}1K{=}$ and v is a variable, then $\forall v\theta$ is a formula of $\mathcal{L}1K{=}$.

The symbol "∀" is called a *universal quantifier*, and is an analogue of "for all"; so $\forall v\theta$ can be read "for all v, θ."

7. If θ is a formula of $\mathcal{L}1K{=}$ and v is a variable, then $\exists v\theta$ is a formula of $\mathcal{L}1K{=}$.

The symbol "∃" is called an *existential quantifier*, and is an analogue of "there exists" or "there is"; so $\exists v\theta$ can be read "there is a v such that θ."

8. That's all folks: all formulas are constructed in accordance with rules (1)– (7).

Clause (8) allows us to do inductions on the complexity of formulas. If a certain property holds of the atomic formulas and is closed under the operations presented in clauses (2)–(7), then the property holds of all formulas. Here is a simple example:

Theorem 3.1 *Every formula of $\mathcal{L}1K{=}$ has the same number of left and right parentheses. Moreover, each left parenthesis corresponds to a unique right parenthesis, which occurs to the right of the left parenthesis. Similarly, each* *right parenthesis corresponds to a unique left parenthesis, which occurs to the left of the given right parenthesis. If a parenthesis occurs between a matched pair of parentheses, then its mate also occurs within that matched pair. In other words, parentheses that occur within a matched pair are themselves matched.*

Proof. By clause (8), every formula is built up from the atomic formulas using clauses (2)–(7). The atomic formulas have no parentheses. Parentheses are introduced only in clauses (3)–(5), and each time they are introduced as a matched set. So at any stage in the construction of a formula, the parentheses are paired off.

We next define the notion of an occurrence of a variable being *free* or *bound* in a formula. A variable that immediately follows a quantifier (as in "$\forall x$" and "$\exists y$") is neither free nor bound. We do not even think of those as occurrences of the variable. All variables that occur in an atomic formula are free. If a variable occurs free in θ or in ψ, then that same occurrence is free in $\neg\theta$, $(\theta \wedge \psi)$, $(\theta \vee \psi)$, and $(\theta \rightarrow \psi)$. The same goes for bound variables. That is, the (unary and binary) connectives do not change the status of variables that occur in them. All occurrences of the variable v in θ are bound in $\forall v\theta$ and $\exists v\theta$. Any *free* occurrences of v in θ are bound by the initial quantifier. All other variables that occur in θ are free or bound in $\forall v\theta$ and $\exists v\theta$, as they are in θ.

For example, in the formula $(\exists x(Axy \vee Bx) \wedge Bx)$, the occurrences of "$x$" in Axy and in the first Bx are bound by the quantifier. The occurrence of "y" and the last occurrence of "x" are free. In $\forall x(Ax \rightarrow \exists xBx)$, the "$x$" in Ax is bound by the initial universal quantifier, while the other occurrence of x is bound by the existential quantifier. The above syntax allows this "double-binding." Although it does not create any ambiguities (see below), we will avoid such formulas, as a matter of taste and clarity.

The syntax also allows so-called vacuous binding, as in $\forall xBc$. This, too, will be avoided in what follows. Some treatments of logic rule out vacuous binding and double binding as a matter of syntax. That simplifies some of the treatments below and complicates others.

Free variables correspond to placeholders, while bound variables are used to express generality. If a formula has no free variables, then it is called a *sentence*. If a formula has free variables, it is called *open*.

3.4 Features of the Syntax

Before turning to the deductive system and semantics, we mention a few features of the language as developed so far. This helps draw the contrast between formal languages and natural languages like English.

We assume at the outset that all of the categories are disjoint. For example, no connective is also a quantifier or a variable, and the non-logical terms are not also parentheses or connectives. Also, the items within each category are distinct. For example, the sign for disjunction does not do double-duty as the negation symbol, and, perhaps more significantly, no two-place predicate is also a one-place predicate.

One difference between natural languages like English and formal languages like $\mathcal{L}1K{=}$ is that the latter are not supposed to have any ambiguities. The policy that the different categories of symbols do not overlap, and that no symbol does double-duty, avoids the kind of ambiguity, sometimes called "equivocation," that occurs when a single word has two meanings: "I've got a bat in my garage" (a piece of baseball equipment or a flying mammal?). But there are other kinds of ambiguity. Consider the English sentence:

> Mohammed is tall and Xenia is smart or James is silly.

It can mean that Mohammed is tall and either Xenia is smart or James is silly, or else it can mean that either both Mohammed is tall and Xenia is smart, or else James is silly. An ambiguity like this, due to different ways to parse the same sentence, is sometimes called an "amphiboly." If our formal language did not have the parentheses in it, it would have amphibolies. For example, there would be a "formula" $A \wedge B \vee C$. Is this supposed to be $((A \wedge B) \vee C)$, or is it $(A \wedge (B \vee C))$? The parentheses resolve what would be an amphiboly.

Can we be sure that there are no other amphibolies in our language? That is, can we be sure that each formula of $\mathcal{L}1K{=}$ can be put together in only one way? The answer is yes, and our next task is to show this.

Let us temporarily use the term "unary marker" for the negation symbol ($\neg$) or a quantifier followed by a variable (e.g., $\forall x, \exists z$).

Lemma 3.2 *Each formula consists of a string of zero or more unary markers followed by either an atomic formula or a formula produced using a binary connective, via one of clauses (3)–(5).*

Proof. We proceed by induction on the complexity of the formula or, in other words, on the number of formation rules that are applied. The Lemma clearly holds for atomic formulas. Let n be a natural number, and suppose that the Lemma holds for any formula constructed from n or fewer instances of clauses (2)–(7). Let θ be a formula constructed from $n+1$ instances. The Lemma holds if the last clause used to construct θ was either (3), (4), or (5). If the last clause used to construct θ was (2), then θ is $\neg\psi$. Since ψ was constructed with n instances of the rule, the Lemma holds for ψ (by the induction hypothesis), and so it holds for θ. Similar reasoning shows the Lemma to hold for θ if the last clause was (6) or (7). By clause (8), this exhausts the cases, and so the Lemma holds for θ, by induction.

Lemma 3.3 *If a formula θ contains a left parenthesis, then it ends with a right parenthesis, which matches the leftmost left parenthesis in θ.*

Proof. This proof also proceeds by induction on the number of instances of (2)–(7) used to construct the formula. Clearly, the Lemma holds for atomic formulas, since they have no parentheses. Suppose, then, that the Lemma holds for formulas constructed with n or fewer instances of (2)–(7), and let θ be constructed with $n + 1$ instances. If the last clause applied was (3)–(5), then the Lemma holds since θ itself begins with a left parenthesis and ends with the matching right parenthesis. If the last clause applied was (2), then θ is $\neg\psi$, and the induction hypothesis applies to ψ. Similarly, if the last clause applied was (6) or (7), then θ consists of a quantifier, a variable, and a formula to which we can apply the induction hypothesis. It follows that the Lemma holds for θ.

Lemma 3.4 *Each formula contains at least one atomic formula.*

Proof. The proof again proceeds by induction on the number of instances of (2)–(7) used to construct the formula.

Theorem 3.5 *Let α, β be nonempty sequences of characters on our alphabet, such that $\alpha\beta$ (i.e., α followed by β) is a formula. Then α is* not *a formula.*

Proof. If α contains a left parenthesis, then the right parenthesis that matches the leftmost left parenthesis in $\alpha\beta$ comes at the end of $\alpha\beta$, and so the matching right parenthesis is in β. So, α has more left parentheses than right parentheses. By Theorem 3.1, α is not a formula. So now suppose that α does not contain any left parentheses. By Lemma 3.2, $\alpha\beta$ consists of a string of zero or more unary markers followed by either an atomic formula or a formula produced using a binary connective, via one of clauses (3)–(5). If the latter formula was produced via one of clauses (3)–(5), then it begins with a left parenthesis. Since α does not contain any parentheses, it must be a string of unary markers. But then α does not contain any atomic formulas, and so by Lemma 3.4, α is not a formula. The only case left is where $\alpha\beta$ consists of a string of unary markers followed by an atomic formula, either in the form $t_1 = t_2$ or $Pt_1 \ldots t_n$. Again, if α just consisted of unary markers, it would not be a formula, and so α must consist of the unary markers that start $\alpha\beta$, followed by either t_1 by itself, $t_1 =$ by itself, or a predicate letter P, and perhaps some (but not all) of the terms $t_1, \ldots, t_n$. In the first two cases, α does not contain an atomic formula, by the policy that the categories do not overlap. Since P is an n-place predicate letter, by the policy that the predicate letters are distinct, P is not an m-place predicate letter for any $m \neq n$. So the part of α that consists of P followed by the terms is not an atomic formula. In all of these cases, then, α does not contain an atomic formula. By Lemma 3.4, α is not a formula.

These theorems are enough to show that there is no amphiboly in our language. Though just an indication of how the proof goes is provided here, the reader can find the complete proof in Shapiro and Kouri Kissel (2020).

Theorem 3.6 *Let θ be any formula of $\mathcal{L}1K{=}$. If θ is not atomic, then there is one and only one among (2)–(7) that was the last clause applied to construct θ. That is, θ could not be produced by two different clauses. Moreover, no formula produced by clauses (2)–(7) is atomic.*

Proof. This proof considers each of clauses (2)–(7) in turn to show there is no amphiboly for any of them. By Clause (8), either θ is atomic or it was produced by one of clauses (2)–(7). Thus, the first symbol in θ must be either a predicate letter, a term, a unary marker, or a left parenthesis. If the first symbol in θ is a predicate letter or term, then θ is atomic. In this case, θ was not produced by any of (2)–(7), since all such formulas begin with something other than a predicate letter or term. If the first symbol in θ is a negation sign "¬," then θ was produced by clause (2), and not by any other clause (since the other clauses produce formulas that begin with either a quantifier or a left parenthesis). Similarly, if θ begins with a universal quantifier, then it was produced by clause (6), and not by any other clause, and if θ begins with an existential quantifier, then it was produced by clause (7), and not by any other clause. The only case left is where θ begins with a left parenthesis. In this case, it must have been produced by one of (3)–(5), and not by any other clause. We only need to rule out the possibility that θ was produced by more than one of (3)–(5). To take an example, suppose that θ was produced by (3) and (4). Then θ is $(\psi_1 \wedge \psi_2)$ and θ is also $(\psi_3 \vee \psi_4)$, where ψ_1, ψ_2, ψ_3, and ψ_4 are themselves formulas. That is, $(\psi_1 \wedge \psi_2)$ is the very same formula as $(\psi_3 \vee \psi_4)$. By Theorem 3.5, ψ_1 cannot be a proper part of ψ_3, nor can ψ_3 be a proper part of ψ_1. So ψ_1 must be the same formula as ψ_3. But then "∧" must be the same symbol as "∨," and this contradicts the policy that each of the symbols are different. So θ was not produced by both Clause (3) and Clause (4). Similar reasoning takes care of the other combinations.

This result is sometimes called "unique readability." It shows that each formula is produced from the atomic formulas via the various clauses in exactly one way. If θ was produced by clause (2), then its *main connective* is the initial "¬." If θ was produced by clauses (3), (4), or (5), then its *main connective* is the introduced "∧," "∨," or "→," respectively. If θ was produced by clauses

(6) or (7), then its *main connective* is the initial quantifier. We apologize for the tedious details. We included them to indicate the level of precision and rigor for the syntax.

4 Deduction

We now introduce a *deductive system*, D, for our languages. As above, we define an *argument* to be a non-empty collection of sentences in the formal language, one of which is designated to be the *conclusion*. If there are any other sentences in the argument, they are its *premises*. We use "Γ," "Γ'," "Γ_1," etc., to range over sets of formulas, and we use the letters "ϕ," "ψ," "θ," uppercase or lowercase, with or without subscripts, to range over single formulas. We write "Γ, Γ'" for the union of Γ and Γ', and "Γ, ϕ" for the union of Γ with $\{\phi\}$.

We write an argument in the form $\langle \Gamma, \phi \rangle$, where Γ is a set of sentences, the premises, and ϕ is a single sentence, the conclusion. Remember that Γ may be empty. We write $\Gamma \vdash \phi$ to indicate that ϕ is deducible from Γ, or, in other words, that the argument $\langle \Gamma, \phi \rangle$ is deducible in D. We may write $\Gamma \vdash_D \phi$ to indicate the deductive system D. We write $\vdash \phi$ or $\vdash_D \phi$ to indicate that ϕ can be deduced (in D) from the empty set of premises. The rules in D are chosen to match inferential relations concerning the English analogues of the logical terminology in the language.

We define the deducibility relation by recursion. We start with a rule of assumptions:

- (As) If ϕ is a member of Γ, then $\Gamma \vdash \phi$.

We thus have that $\{\phi\} \vdash \phi$; each premise follows from itself. We next present two clauses for each connective and quantifier. The clauses indicate how to "introduce" and "eliminate" sentences in which each symbol is the main connective.

First, recall that "$\wedge$" is an analogue of the English connective "and." Intuitively, one can deduce a sentence in the form $(\theta \wedge \psi)$ if one has deduced θ and one has deduced ψ. Conversely, one can deduce θ from $(\theta \wedge \psi)$, and one can deduce ψ from $(\theta \wedge \psi)$:

- ($\wedge$I) If $\Gamma_1 \vdash \theta$ and $\Gamma_2 \vdash \psi$, then $\Gamma_1, \Gamma_2 \vdash (\theta \wedge \psi)$.
- ($\wedge$E) If $\Gamma \vdash (\theta \wedge \psi)$, then $\Gamma \vdash \theta$; and if $\Gamma \vdash (\theta \wedge \psi)$, then $\Gamma \vdash \psi$.

The name "($\wedge$I)" stands for "$\wedge$-introduction"; "($\wedge$E)" stands for "$\wedge$-elimination."

Since the symbol "$\vee$" corresponds to the English "or," $(\theta \vee \psi)$ should be deducible from θ, and $(\theta \vee \psi)$ should also be deducible from ψ: