A Finitely Axiomatized Formalization of Predicate Calculus with Equality

Note: This is a preprint of Megill, "A Finitely Axiomatized Formalization of Predicate Calculus with Equality," *Notre Dame Journal of Formal Logic,* 36:435-453, 1995.

The paper as published has the following errata that have been corrected in this preprint.

- On p. 439, line 7, "(Condensed detachment)" should be followed by a reference to a footnote, "The arrays start at index $i = 1$. In Step 3, i is increased *before* each comparison is made."
- On p. 446, line 17, "unnecessary" is misspelled.
- On p. 448, 2nd line from bottom, "that in Section 8." should be followed by "In addition, S3′ has the following stronger property."
- On p. 449, line 22, "u_i" should be "u_1".
- On p. 450, line 27, "Dpq" should be "Dqp".
- On p. 451, line 2, "shorter proof string" should be followed by a reference to a footnote, "Found by the OTTER theorem prover [20]."

In August 2017, Tony Häger identified and provided the proofs for two missing lemmas needed for the Substitution Theorem proof, and also found some shorter proofs for other lemmas. The necessary corrections have not yet been incorporated in this preprint but can be found in the discussion starting at `https://groups.google.com/d/msg/metamath/t2G8X-dvTbA/TqfxeO1sAwAJ`

A Finitely Axiomatized Formalization of Predicate Calculus with Equality

July 7, 1995

Abstract

We present a formalization of first-order predicate calculus with equality which, unlike traditional systems with axiom schemata or substitution rules, is finitely axiomatized in the sense that each step in a formal proof admits only finitely many choices. This formalization is primarily based on the inference rule of condensed detachment of C. A. Meredith. The usual primitive notions of free variable and proper substitution are absent, making it easy to verify proofs in a machine-oriented application. Completeness results are presented. The example of Zermelo-Fraenkel set theory is shown to be finitely axiomatized under the formalization. The relationship with resolution-based theorem provers is briefly discussed. A closely related axiomatization of traditional predicate calculus is shown to be complete in a strong metamathematical sense.

1 Introduction

We define a formal theory to be *finitely axiomatized* if each step of a formal proof in the theory admits only finitely many choices. This definition implies that the underlying logic is finitely axiomatized; it is stricter than the usual definition which requires only that the number of nonlogical axioms of the theory be finite.

Traditional axiom systems of first-order predicate calculus are usually presented with axiom schemata each of which represents infinitely many axioms (e.g. [4, p. 73]). Throughout this paper we assume familiarity with such a system and refer to it as *traditional predicate calculus*. Church [2, pp. 218–219] presents a formalization of predicate calculus with a finite number of axioms together with substitution rules. However, a substitution rule is really a rule schema admitting an infinite number of choices, so Church's system is not finitely axiomatized in our terminology.

The finiteness of the number of nonlogical axioms in a theory is often considered philosophically or aesthetically desirable, as in, for example, NBG (von

Neumann-Bernays-Gödel) set theory [10, pp. 173–219]. However, the underlying logic usually tends to be viewed only in terms of schemata of infinite axioms. Kleene [8, p. 140] writes that whether we set up propositional calculus with axiom schemata or with particular axioms and a substitution rule, "the rules of inference must have the character of schemata, i.e. they must employ metamathematical variables, since infinitely many applications have to be provided for." Tarksi and Givant [18, p. 7], state that "[the] set of logical axioms is necessarily infinite" in a formalization of predicate calculus.

We shall describe a new formalization of predicate calculus with equality that is finitely axiomatized when the number of nonlogical symbols in the language is finite. Finite axiomatization is achieved by treating what are ordinarily thought of as metavariables as the primitive variables of the system and providing inference rules to manipulate these directly. We then show how to map a subset of the resulting "metatheorems" directly into the theorems of traditional predicate calculus and show that this mapping is complete. (However, not all formulas of our system that are true when interpreted as metatheorems are provable; see Remark 2 in Section 9.)

Our formalization uses C. A. Meredith's inference rule of *condensed detachment* (Rule **D**) [11] in place of modus ponens and substitution. Any axiom system with Rule **D** as its rule of inference is finitely axiomatized in our sense because at any point in a proof there are only finitely many earlier steps to which Rule **D** can be applied, and the outcome of Rule **D** is unique (up to renaming variables). However the underlying axiom system has to have a certain minimum strength in order for the system to be complete, in the sense of being able to prove all possible substitution instances of theorems. Some properties sufficient to achieve this **D**-*completeness* for weak implicational systems are discussed in [5] and [9].

The remaining remarks in this section assume the reader is familiar with resolution and related techniques in the field of automated reasoning.

An inference rule related to Rule **D** is the *resolution principle* of J. A. Robinson [16], commonly used in automated theorem proving. A deductive system of logic whose inference rules are resolution together with the related rules of factoring and paramodulation can be considered finitely axiomatized in our sense and can produce clauses (theorems) that correspond to Skolemizations of theorems of predicate calculus with equality. But such a system is not "deduction complete" so that, for example, Gödel's completeness theorem fails to hold [21].

However, it is possible to use resolution with our system S1 below to achieve deduction completeness in an indirect way. Kalman [7] showed that Rule **D** can be subsumed within resolution by treating formulas of logic as terms and introducing a provability predicate, a method now often used (in refutation systems) to find proofs in fragments of propositional calculus [20, pp. 355 and 480–482]. This method represents Rule **D** with the clause $-P(x)|-P(i(x,y))|P(y)$ where P is the provability predicate and i represents the binary connective $\rightarrow$ in system

S1. The other inference rule we will need for system S1, *condensed generalization*, also can be subsumed within resolution with the clause $-P(x)|P(a(y,x))$ where a represents the binary connective $\forall$ in system S1. Because resolution-style inferences preserve the positions of arguments, individual and propositional variables will not mix and thus will not allow us to prove ill-formed formulas.

Resolution-based theorem provers typically have other inference rules involving substitution, and we must ensure that these also will not mix variables in the presence of a candidate theorem that qualifies as a wff (as defined below). If we are uncertain about the substitution rules used by a theorem prover, we can restore confidence in its output by verifying that each step of the generated proof is a wff. Roughly, any resolution-based prover suitable for assisting **D**-completeness proofs (e.g. [12]) is sound for system S1. However a brief experiment showed that the theorem prover described in [20], which is not suitable in this sense, did correctly prove general cases of our lemmas L1–L22 whenever it was able (on a small computer) to find a proof (from which the exact lemmas follow by the Substitution Theorem below).

2 The System S1

To simplify our notation, we shall restrict our study to a subset of predicate calculus that has equality, one additional binary predicate symbol, and no function or constant symbols. This subset suffices for set theory, and the choice of the symbol $\in$ for the additional binary predicate is not accidental, but the axioms apply to any binary predicate. It is possible to add an arbitrary finite number of n-place predicate symbols to the system in a straightforward fashion and, using methods described in [8, pp. 405–420], to add constants and function symbols by means of definitions.

We now describe S1, the main system of our discourse. The undefined symbols of S1 are represented by an infinite number of variables in sequence a, b, c,...; a unary connective $\neg$; and binary connectives $\rightarrow$, $\forall$, $=$, $\in$. A *formula* is a string of these symbols. The axioms of S1 are the following formulas.

(A1) $$\rightarrow a \rightarrow ba$$

(A2) $$\rightarrow\rightarrow a \rightarrow bc \rightarrow\rightarrow ab \rightarrow ac$$

(A3) $$\rightarrow\rightarrow \neg a\neg b \rightarrow ba$$

(A4) $$\rightarrow \forall a \rightarrow \forall abc \rightarrow \forall ab\forall ac$$

(A5) $$\rightarrow \forall abb$$

(A6) $$\rightarrow \forall a\forall bc\forall b\forall ac$$

(A7) $$\rightarrow \neg a\forall b\neg\forall ba$$

(A8) $$\rightarrow= ab \rightarrow= ac = bc$$

(A9) $$\rightarrow \neg\forall a = ab \rightarrow \neg\forall a = ac \rightarrow= bc\forall a = bc$$

(A10) $$\rightarrow \forall a \rightarrow = ab \forall acc$$

(A11) $$\rightarrow \forall a = ab \rightarrow \forall ac \forall bc$$

(A12) $$\rightarrow = ab \rightarrow \in ac \in bc$$

(A13) $$\rightarrow = ab \rightarrow \in ca \in cb$$

(A14) $$\rightarrow \neg \forall a = ab \rightarrow \neg \forall a = ac \rightarrow \in bc \forall a \in bc$$

The inference rules of S1 are *condensed detachment* (Rule **D**) and *condensed generalization* (Rule **G**). They are analogous to modus ponens and generalization in traditional predicate calculus, but we shall defer their precise definition until the next section.

Unless otherwise stated, we shall use F, G, and H (possibly with subscripts) as metavariables ranging over formulas, and u and v as metavariables ranging over variables.

To facilitate the description of the rules of S1, we have displayed its axioms in *Polish prefix* notation, in which a variable is an atomic *primitive formula*, and if F and G are primitive formulas, so are $\neg F$, $\rightarrow FG$, $\forall FG$, $= FG$, and $\in FG$.

A *proof* in S1 is a finite sequence of *theorems*, each of which is an axiom, the result of Rule **D** applied to two previous theorems in the sequence (if the result exists), or the result of Rule **G** applied to a previous theorem in the sequence. The *result of a proof* is the last theorem in the sequence. A **D**-*derivation* is a proof from a finite number of axioms that uses **D** and **G** as the only rules; thus all proofs in system S1 are **D**-derivations.

To avoid ambiguity, we shall sometimes call an arbitrary variable of system S1 a *primitive variable*. We define an *individual variable* as a (primitive) variable that occurs as the first argument of a $\forall$ connective or as either argument of an $=$ or $\in$ connective. A variable in any other position is defined as a *propositional variable*.

We define an atomic *well-formed formula* (*wff*) as either a (propositional) variable or a primitive formula of the form $= uv$ or $\in uv$; and if F and G are wffs, so are $\neg F$, $\rightarrow FG$, and $\forall uF$, provided that no variable becomes both a propositional variable and an individual variable. It may be observed that each axiom of S1 is a wff.

The variables in a formula are *normalized* if they occur in alphabetic order a, b, c,... by first appearance. The axioms and rules of S1 are such that all theorems have normalized variables.

The connectives $=$ and $\in$ are *predicate symbols*. The connectives $\neg$, $\rightarrow$, and $\forall$ are *logical connectives*. The predicate symbol $\in$ is a *nonlogical symbol*.

3 The Rules D and G

The rule of *condensed detachment*, which we shall call *Rule* **D**, was introduced by C. A. Meredith [11] as a method for abbreviating proofs in propositional calculus. In traditional propositional calculus, the result of applying modus ponens (with some appropriate substitutions), if it can be applied, to theorems of the form $\rightarrow FG$ and H is an infinite set of substitution instances of G. From this set Rule **D** picks a theorem that is *most general* in the sense that all other theorems in the set are substitution instances of it. To make it unique, the most general theorem that is picked is the one with its variables normalized according to some convention. We denote this most general theorem by $\mathbf{D}(\rightarrow FG, H)$ and say that it is the result of *detaching* H from $\rightarrow FG$. Rigorous treatments of Rule **D** are provided in [5] and [7], and an informal tutorial is presented in [22, pp. 2–6]. A specific example is given in the Appendix of this paper.

We extend the use of Rule **D** to system S1, treating all connectives as if they were propositional connectives. It is important to note that in system S1, unlike propositional calculus, the general substitution of a variable with a wff is not a defined or derived rule, nor will such a substitution necessarily result in a valid theorem. We shall show later that when operating on the axioms of S1, Rule **D** will perform only acceptable substitutions and, less obviously, can perform all possible acceptable substitutions. This is the essential feature that allows the predicate calculus to be complete yet still finitely axiomatized, because each application of Rule **D** (or Rule **G**) results in a unique theorem. The nature of Rule **D** is such that when it is applied to two wffs, the result, if it exists, is a wff; in particular, Rule **D** will never mix individual and propositional variables.

The following algorithm from Peterson [15], with the addition of step 10, generates $\mathbf{D}(\rightarrow FG, H)$ or shows that it does not exist. A *subformula* is the shortest sequence of symbols that begins at an indicated point in a formula and satisfies the definition of a primitive formula. F is the subformula beginning at the second symbol in $\rightarrow FG$. To find $\mathbf{D}(\rightarrow FG, H)$, we represent formulas F, G, and H as strings of nonzero integers stored left-justified in arrays A, B, and C such that (for example) the variables are represented by positive integers and the connectives by negative integers. A zero represents the end of a string.

Algorithm D (Condensed detachment)[1] *Step 1.* Renumber the variables in C so that it has no variables in common with A and B. *Step 2.* Set i to 0. *Step 3.* Increase i by 1 until $A[i] \neq C[i]$ or $C[i] = 0$. *Step 4.* If $C[i] = 0$ then go to 10. Otherwise continue. *Step 5.* If $C[i]$ is a variable then set m to $C[i]$ and place the subformula beginning with $A[i]$ in array E. Go to 8. Otherwise continue. *Step 6.* If $A[i]$ is a variable then set m to $A[i]$ and place the subformula beginning with $C[i]$ in array E. Go to 8. Otherwise continue. *Step 7.* Terminate the algorithm. $\mathbf{D}(\rightarrow FG, H)$ does not exist. *Step 8.* If m occurs in array E then go to 7. Otherwise continue. *Step 9.* Substitute the

[1] The arrays start at index $i = 1$. In Step 3, i is increased *before* each comparison is made.

content of array E for each occurrence of m throughout arrays A, B, and C. Go to 3. *Step 10.* (*Normalization*) Renumber the variables in B so that they occur, by first appearance, in alphabetic order a, b, c,.... Terminate the algorithm. The content of array B is $\mathbf{D}(\rightarrow FG, H)$.

Condensed generalization or *Rule* **G** quantifies a theorem F with a variable not appearing in the theorem and normalizes the result. The following algorithm performs this rule.

Algorithm G (Condensed generalization) *Step 1.* Change each variable in F to the next variable in the language's list of variables, so that a becomes b, b becomes c, etc. *Step 2.* Preface F with "$\forall a$". Terminate the algorithm.

4 The System S2

We next define S2, an axiomatization that represents an intermediate step between S1 and traditional predicate calculus. We shall later show that S1 and S2 are equivalent.

S1 and S2 differ only in their rules. Except for notation, the axioms of system S2 are the same as those of S1. The individual and propositional variables of S1 are replaced with distinguished groups of symbols and the notation is changed from Polish prefix to a more conventional one. Implicitly, each theorem of S2 is a metamathematical representation of a theorem in the normalized notation of S1, even though the axioms of S2 do not explicitly exhibit nor the rules explicitly require variable normalization. We allow the theorems of S2 to contain unnormalized variables only as a metamathematical convenience. Henceforth, we shall restrict the metavariables u and v to range over individual variables (and not propositional variables) unless otherwise stated.

The symbols of S2 are represented by propositional variables P, Q, R, S,...; individual variables x, y, z,...; unary connective $\neg$; binary connectives $\rightarrow$, $\forall$, $=$, $\in$; and parentheses (,).

In system S2, an atomic wff is either a propositional variable or an expression of the form $u = v$ or $u \in v$; and if F and G are wffs, so are $\neg F$, $(F \rightarrow G)$, and $\forall uF$. For readability we omit the outermost parentheses when writing formulas. The axioms of S2 follow, separated into related groups:

(Axioms for propositional calculus)

(B1) $$P \rightarrow (Q \rightarrow P)$$

(B2) $$(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$$

(B3) $$(\neg P \rightarrow \neg Q) \rightarrow (Q \rightarrow P)$$

(Axioms for pure predicate calculus)

(B4) $$\forall x(\forall xP \rightarrow Q) \rightarrow (\forall xP \rightarrow \forall xQ)$$

(B5) $$\forall xP \rightarrow P$$

(B6) $$\forall x \forall y P \rightarrow \forall y \forall x P$$

(B7) $$\neg P \rightarrow \forall x \neg \forall x P$$

(Axioms for equality and substitution)

(B8) $$x = y \rightarrow (x = z \rightarrow y = z)$$

(B9) $$\neg \forall x x = y \rightarrow (\neg \forall x x = z \rightarrow (y = z \rightarrow \forall x y = z))$$

(B10) $$\forall x (x = y \rightarrow \forall x P) \rightarrow P$$

(B11) $$\forall x x = y \rightarrow (\forall x P \rightarrow \forall y P)$$

(Axioms for a binary predicate)

(B12) $$x = y \rightarrow (x \in z \rightarrow y \in z)$$

(B13) $$x = y \rightarrow (z \in x \rightarrow z \in y)$$

(B14) $$\neg \forall x x = y \rightarrow (\neg \forall x x = z \rightarrow (y \in z \rightarrow \forall x y \in z))$$

The inference rules of system S2 are the following. (1) *Modus ponens*: From F and $F \rightarrow G$, infer G. (2) *Generalization*: From F, if u is any individual variable, infer $\forall u F$. (3) *Substitution for propositional variables*: From F, infer a new formula obtained by replacing all occurrences of some propositional variable in F with any wff. (4) *Substitution for individual variables*: From F, infer a new formula obtained by replacing all occurrences of some individual variable in F with any other individual variable.

In the modus ponens inference, F is called the *minor premise* and $F \rightarrow G$ the *major premise*. In the major premise, F is called the *antecedent* and G the *consequent*. We say that F is *detached* from $F \rightarrow G$ to result in G (although usually we shall use the word "detach" in the sense of condensed detachment defined above). We may use the term "antecedent" somewhat informally; for example in $F \rightarrow (G \rightarrow H)$, F and G may both be called antecedents.

It will be convenient to have available other logical connectives defined in terms of the primitive connectives; we define $F \vee G$ (*disjunction*), $F \wedge G$ (*conjunction*), $F \leftrightarrow G$, and $\exists u F$ as abbreviations for $\neg F \rightarrow G$, $\neg(F \rightarrow \neg G)$, $\neg((F \rightarrow G) \rightarrow \neg(G \rightarrow F))$, and $\neg \forall u \neg F$ respectively.

Unlike S1, S2 is not finitely axiomatized because of its substitution rules. However, S2 is interesting in its own right because the axioms have no verbal restrictions on variables, i.e. there are no primitive notions of free, bound, or distinct individual variables, nor are there complex rules for proper substitutions. This property is essential for compatibility with Rule **D** of S1, but as a side benefit the inherent simplicity of S2 can make it easy to study as a formal system.

Because the wffs of S2 are the same as those of S1 except for notation, we shall usually represent wffs of S1 in the more standard notation of S2. We shall also interchangeably refer to axioms A1 through A14 and B1 through

B14, whichever are more convenient for the situation at hand. It should be emphasized, however, that the wffs of S2 are metamathematical representations of those of S1 and at the primitive level there is no distinction between individual and propositional variables in S1. After we show that S1 and S2 are equivalent, we shall interchangeably refer to these two systems.

5 Equivalence of Systems S1 and S2

We say that two formal systems are *equivalent* if any theorem that can be proved in one system can also be proved in the other (except for a possible difference of notation). In the case of systems S1 and S2, our main task is to prove from S1 the substitution rules of S2. The rest of the proof follows as easy corollaries.

Theorem 5.1 (Substitution Theorem) *Rules* **D** *and* **G** *of system* S1 *generate exactly those substitution instances defined by the substitution rules of system* S2.

Proof: First, we show that Algorithm **D** can generate *only* those substitution instances defined by the substitution rules of system S2. By inspection of Algorithm **D** and the axioms of S1, we notice that only individual variables (never wffs) will be substituted for individual variables. This follows from the fact that the axioms are wffs, so that no wff appears as an argument of an $=$ or $\in$ connective nor as the first argument of a $\forall$ connective. Similarly, we notice that only wffs (never individual variables) will be substituted for propositional variables. Thus Rule **D** will never violate the substitution rules of system S2.

Similarly, Algorithm **G** will generate only acceptable substitution instances of the generalization rule of S2.

It remains to be shown that Rules **D** and **G** are complete, i.e. that they can derive *all* instances of the substitution rules of system S2.

Assume that a proof exists in system S2 for some theorem F_s. We want to show that we can prove this theorem using the axioms and rules of system S1.

First we convert the proof in S2 to a proof in S1 by deleting all applications of the substitution rules, replacing modus ponens with Rule **D** and replacing generalization with Rule **G**. The result of the new proof will be a most general theorem F_g of which the desired theorem F_s is a substitution instance.

Rule **D** has the following property. If F_s is any substitution instance of a theorem F_g, then $\mathbf{D}(\rightarrow F_s F_s, F_g)$ (i.e. F_g detached from $F_s \rightarrow F_s$) is F_s. This is easy to see by examining Algorithm **D**.

We shall show that any formula of the form $F_s \rightarrow F_s$, where F_s is a wff, is **D**-derivable, i.e. can be proved in system S1. Then by applying the property of Rule **D** just mentioned, we can **D**-derive F_s when it is a substitution instance of some theorem F_g. This will complete our proof of the Substitution Theorem.

First we construct a proof of a theorem of the form $F_v \rightarrow (F_d \rightarrow F_d)$. F_d is a wff identical to F_s except that all appearances of the propositional variables and individual variables in F_d will be distinct, i.e. each variable will appear only

once. F_v is a disjunction of all of the propositional variables contained in F_d and also of formulas of the form $\forall u G$, one for each individual variable u contained in F_d (and G is a dummy "place holder" propositional variable not in F_d nor elsewhere in F_v).

We prove $F_v \to (F_d \to F_d)$ by induction on the number of connectives in the wff F_d (i.e. in F_s). The induction basis is one of the **D**-derivable lemmas

$$(L1) \qquad P \to (P \to P)$$

$$(L2) \qquad (\forall x P \vee \forall y Q) \to (x = y \to x = y)$$

$$(L3) \qquad (\forall x P \vee \forall y Q) \to (x \in y \to x \in y)$$

corresponding to the atomic wffs P, $x = y$, and $x \in y$. (The proofs of all lemmas are given in the Appendix.) Note that in L2 and L3, P and Q are dummy place holder propositional variables mentioned above, and their purpose is to capture x and y so that x and y can be manipulated in a manner similar to propositional variables.

For the induction hypothesis, we assume that all theorems with fewer connectives than $F_v \to (F_d \to F_d)$ and of that form can be proven. F_d, if not atomic, must be of the form $\neg G$, $G \to H$, or $\forall u G$ by the definition of a wff. We prove $F_v \to (F_d \to F_d)$ by detaching from one of the **D**-derivable lemmas

$$(L4) \qquad (P \to (Q \to Q)) \to (P \to (\neg Q \to \neg Q))$$

$$(L5) \qquad (P \to (Q \to Q)) \to ((R \to (S \to S)) \to ((P \vee R) \to ((Q \to S) \to (Q \to S))))$$

$$(L6) \qquad (P \to (Q \to Q)) \to ((P \vee \forall x R) \to (\forall x Q \to \forall x Q))$$

corresponding to the connectives $\neg$, $\to$, and $\forall$ respectively. In L6, R is a dummy propositional variable that captures x.

At this point we have completed constructing a proof of $F_v \to (F_d \to F_d)$. Each variable in F_d will appear only once, whereas in F_s some variables will probably appear more than once. We must transform F_d to F_s by forcing some of the distinct propositional variables and distinct individual variables in F_d to become identical to each other. Pick two such propositional variables G and H. If F_v has more than two disjuncts, we detach $F_v \to (F_d \to F_d)$ repeatedly from the **D**-derivable lemmas

$$(L7) \qquad ((P \vee Q) \to R) \to ((Q \vee P) \to R)$$

$$(L8) \qquad (((P \vee Q) \vee R) \to S) \to ((P \vee (Q \vee R)) \to S)$$

to make G and H become the leftmost disjuncts of F_v, so that F_v is of the form $(G \vee H) \vee F$ where F is the rest of the disjunction. (We are implicitly assuming that we are keeping track of the variables metamathematically, since the actual subtheorems in the notation of system S1 have normalized variables after each

application of Rule **D**.) Next we detach from one of the **D**-derivable lemmas

$$(\text{L9}) \qquad ((P \vee P) \to Q) \to ((P \vee P) \to Q)$$

$$(\text{L10}) \qquad (((P \vee P) \vee Q) \to R) \to (((P \vee P) \vee Q) \to R)$$

(L9 if F_v has only two disjuncts, L10 otherwise) to force the two distinct propositional variables G and H to match each other. The result is a theorem of the form $F'_v \to (F'_d \to F'_d)$ in which the propositional variables G and H are now identical.

To make individual variables identical to each other, we perform the same kinds of manipulations with L7 and L8, except that we move disjuncts of the form $\forall uG$ and $\forall vH$ to the leftmost positions. Detaching from L9 or L10 will force u and v to match each other; G and H in this case will also become identical, but this is irrelevant because they are dummy propositional variables.

We repeat the above steps for all distinct variables in F_d that must be made identical to obtain a theorem of the form $F''_v \to (F_s \to F_s)$. We detach this from the **D**-derivable lemma

$$(\text{L11}) \qquad (P \to (Q \to Q)) \to (Q \to Q)$$

to discard the antecedent and finally obtain $F_s \to F_s$.

A simple example is helpful to understand these steps. Suppose we have proved $P \to P$ in system S1, and we want to prove the substitution instance $\neg P \to \neg P$. Let F_g be $P \to P$ and F_s be $\neg P \to \neg P$. First we construct a proof for $F_s \to F_s$, i.e. $(\neg P \to \neg P) \to (\neg P \to \neg P)$. We start with Lemma L1, $P \to (P \to P)$. We detach this from L4 to obtain $P \to (\neg P \to \neg P)$, which we then detach twice from L5 to obtain $(P \vee Q) \to ((\neg P \to \neg Q) \to (\neg P \to \neg Q))$. We must make P and Q identical. Since there are only two propositional variables, we do not need L7 or L8. Detaching from L9, we obtain $(P \vee P) \to ((\neg P \to \neg P) \to (\neg P \to \neg P))$. Detaching from L11, we obtain $(\neg P \to \neg P) \to (\neg P \to \neg P)$, which is the desired $F_s \to F_s$. Detaching $P \to P$ from $(\neg P \to \neg P) \to (\neg P \to \neg P)$, we finally obtain $\neg P \to \neg P$.

(End of proof of Substitution Theorem.) □

The Substitution Theorem shows that the substitution rules of S2 can be derived from the axioms and rules of S1. As an easy corollary, the modus ponens rule of S2 is a special case of Rule **D** of S1, and the generalization rule of S2 follows from Rule **G** of S1 and the Substitution Theorem. Conversely, since Rule **D** can be used as a rule of inference in a substitution and detachment system [7], Rule **D** of S1 follows from modus ponens and the substitution rules of S2, and Rule **G** of S1 follows as a special case of the generalization rule of S2. The axioms of S1 and S2 are identical except for notation. It follows that S1 and S2 have the same power of proof.

Having proved the Substitution Theorem, we shall make use of it implicitly henceforth in order to shorten proofs. Thus in the Appendix, a formal proof of

any of the remaining lemmas in this paper may yield a more general case of the lemma.

6 The System S3

Tarski [17] presents a simplified axiom system for traditional predicate calculus. His system shares some similarities with our S2 above, and it will be convenient to prove the completeness of S2 (and hence S1) by deriving from S2 that fragment of Tarski's system which contains only the predicate symbols $=$ and $\in$. We shall call this fragment system S3. (For simplicity we are concerned with completeness of systems with only these two predicate symbols; extension to an arbitrary finite number of predicate symbols is straightforward.) We denote the infinite set of (individual) variables of S3 arranged in sequence $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$,.... If u and v are variables, then $u = v$ and $u \in v$ are atomic wffs; and if F and G are wffs, so are $\neg F$, $F \to G$, and $\forall u F$. The following are the axiom schemata of S3, where F, G, and H are wffs:

(C1) $$(F \to G) \to ((G \to H) \to (F \to H))$$

(C2) $$(\neg F \to F) \to F$$

(C3) $$F \to (\neg F \to G)$$

(C4) $$\forall u(F \to G) \to (\forall u F \to \forall u G)$$

(C5) $F \to \forall u F$, where u is not among the set of variables occurring in F

(C6) $\neg\forall u \neg u = v$, where u and v are distinct variables

(C7) $u = v \to (F \to G)$, where F is any atomic wff in which u occurs, and G is obtained from F by replacing a single occurrence of u with v

The inference rules of S3 are modus ponens and generalization.

7 Mapping Formulas from S1 into S3

We define a *distinctor* as a formula in S1, S2, or S3 of the form $\neg\forall u u = v$ where u and v are distinct variables. We define as *propositionless* a formula of S1 (or S2) containing only individual variables. We recall that the variables of S1 are arranged in sequence a, b, c,... and those of S3 in sequence $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$,.... In this section we shall use non-Polish notation when displaying formulas of system S1.

Certain axioms of S3 require that some individual variables be distinct from one another, a requirement absent from S1; in particular, the Substitution Theorem permits arbitrary substitutions of individual variables for individual variables in S1. Therefore we must represent the formulas of S3 indirectly in S1 by means of a suitable mapping. The completeness of S1 will then be proved by showing that the set of propositionless theorems of S1 maps *onto* the set of

theorems of S3. (We shall not be concerned with those theorems of S1 that have propositional variables since any such theorem corresponds to a theorem schema, not a particular theorem, of system S3.)

The mapping we shall use in the completeness proof of S1 is the *method of distinctor elimination*. This mapping requires that we sacrifice soundness in the one-element domain because, as we shall see, the theorem $\neg\forall aa = b \rightarrow \neg\forall aa = b$ in S1 maps to the formula $\neg\forall\mathbf{xx} = \mathbf{y}$ in S3 which is false in an interpretation of S3 with a one-element domain. However, we have chosen to use this mapping because of its simplicity. (It is possible to devise other mappings that are sound in all non-empty domains. S1 is complete in the one-element domain provided that we add to any theory with that domain the nonlogical axiom $a \rightarrow \forall ba$.)

The method of distinctor elimination makes use of the following fact. A distinctor $\neg\forall uu = v$ in S3 is true in all multiple-element domains (it also a theorem of set theory). It can therefore be detached when used as an antecedent of a theorem of S3, which under this mapping is implicitly extended to exclude the one-element domain. Specifically, this mapping first replaces the variables a, b, c,... in a theorem of S1 with the variables $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$,... of S3 on a one-to-one basis. It then discards the theorem's antecedent (if there is one) when the antecedent is a distinctor or a conjunction of distinctors. For example, the theorems $\exists aa = b$, $(\neg\forall aa = b \wedge \neg\forall aa = c) \rightarrow \exists aa = b$, $\neg\forall aa = b \rightarrow (\neg\forall aa = c \rightarrow \exists aa = b)$, and $\neg\forall aa = a \rightarrow \exists aa = b$ are mapped to $\exists\mathbf{xx} = \mathbf{y}$, $\exists\mathbf{xx} = \mathbf{y}$, $\neg\forall\mathbf{xx} = \mathbf{z} \rightarrow \exists\mathbf{xx} = \mathbf{y}$, and $\neg\forall\mathbf{xx} = \mathbf{x} \rightarrow \exists\mathbf{xx} = \mathbf{y}$ respectively (note that $\neg\forall\mathbf{xx} = \mathbf{x}$ is not a distinctor). The completeness proof of S2 (and thus S1) will show that any theorem of S3 can be proved in S2 provided we allow the theorem to be prefixed with a possible antecedent consisting of a distinctor or a conjunction of distinctors.

Distinctors are primarily intended to specify those pairs of variables in the theorem that must be distinct. If the Substitution Theorem is used to replace the two variables in a distinctor with a single variable, the distinctor becomes the false formula $\neg\forall uu = u$, and the theorem will remain valid because $\neg\forall uu = u$ is conjoined to its antecedent. After making such a substitution, the method of distinctor elimination will not discard $\neg\forall uu = u$ thus ensuring validity of the theorem in S3.

Distinctors have two other roles. First, the variables in any theorem of S1 are normalized, whereas S3 permits any permutation of variables in a theorem. To achieve an arbitrary permutation of variables in the representation in S3, the order of the distinctors can be rearranged as needed and dummy distinctors added so that the part of the theorem after the conjunction of distinctors has its variables appear with the desired permutation. For example, $\neg\forall aa = b \rightarrow \exists bb = a$ becomes $\exists\mathbf{yy} = \mathbf{x}$ and $(\neg\forall aa = b \wedge \neg\forall aa = c) \rightarrow \exists cc = a$ becomes $\exists\mathbf{zz} = \mathbf{x}$. The rearranging and adding of distinctors is done using simple tautologies, the Substitution Theorem and the following lemma:

(L12) $$\neg\forall aa = b \rightarrow \neg\forall bb = a$$

Second, a result of Andréka (proved in [14]) shows that if we restrict S3 to the fragment containing only a finite number n of variables ($n > 2$) then infinitely many wff metavariables are required for a complete axiomatization. Since the axiomatization of S3 has finitely many wff metavariables, it follows that *dummy variables*, distinct from the variables in the theorem to be proved, must sometimes be introduced during the course of a proof in S3. In system S2 (and thus S1), these dummy individual variables must remain embedded in the result of the proof, for otherwise we could simply substitute for them, throughout a proof, some other individual variable that occurs in the result, then restate the proof in the language of S3, contrary to [14]. However, propositionless theorems can be proved so that the dummy variables appear only in the conjunction of distinctors that forms the antecedent of the theorem. This fact falls out of the method we use to construct a proof for axiom C5 in the next section, C5 being the only axiom that is not valid unless certain variables are distinct. Distinctors can thus serve to collect the dummy variables and discard them when the theorem is mapped into S3.

8 Completeness and Consistency of System S2

We now focus on the completeness of system S2, from which the completeness of S1 will follow because of its equivalence with S2. In this discussion, we may implicitly assume that distinctors are being used as described above to indicate distinct variable restrictions, eliminate dummy variables, and force specific permutations of variables in the theorems of S3.

In this section we assume that system S3 is consistent and complete. The proofs are found in [17] and [6].

First we show that S2 is consistent. The axioms of S2 are easily seen to be metatheorems of S3 when the propositional and individual variables of S2 are interpreted as metavariables ranging over wffs and variables of S3. It is also easy to verify that the axioms remain metatheorems when subjected to the substitution rules, i.e. that there are no distinct variable restrictions on the axioms. The rules of modus ponens and generalization are the same as in system S3. These two rules preserve soundness. Finally, it is easy to see that a substitution rule applied to the result of another rule can be eliminated by making appropriate substitutions at earlier steps in a proof, so that an equivalent proof can be obtained in which substitutions apply only to axioms. Therefore system S2 is consistent.

It remains to be shown that system S2 is complete. We shall do this by deriving from S2 the axioms and rules of system S3.

Axiom schemata C1 through C4 correspond to all substitution instances of the following four lemmas of S2, whose proofs are in the Appendix.

(L13) $$(P \to Q) \to ((Q \to R) \to (P \to R))$$

(L14) $$(\neg P \to P) \to P$$

(L15) $$P \to (\neg P \to Q)$$

(L16) $$\forall x(P \to Q) \to (\forall x P \to \forall x Q)$$

Axiom schema C7 corresponds to all substitution instances of the following three axioms and one lemma of S2.

(B8) $$x = y \to (x = z \to y = z)$$

(L17) $$x = y \to (z = x \to z = y)$$

(B12) $$x = y \to (x \in z \to y \in z)$$

(B13) $$x = y \to (z \in x \to z \in y)$$

Axiom schema C6 has the unnecessary verbal restriction "where u and v are distinct variables." In S2, we can prove the following more general restrictionless lemma.

(L18) $$\neg\forall x \neg x = y$$

Axiom schema C5 is proved as a metatheorem of S2 by induction on the number of connectives in the wff F in $F \to \forall u F$. The basis for the induction are the following two axioms of S2 along with those substitution instances in which x remains distinct from y and z.

(B9) $$\neg\forall x x = y \to (\neg\forall x x = z \to (y = z \to \forall x y = z))$$

(B14) $$\neg\forall x x = y \to (\neg\forall x x = z \to (y \in z \to \forall x y \in z))$$

Here, the antecedents of the form $\neg\forall u u = v$ are distinctors specifying (when mapping to S3) that u and v be distinct variables. Thus B9 and B14 state, in effect, "$y = z \to \forall x y = z$ where x does not occur in $y = z$" and "$y \in z \to \forall x y \in z$ where x does not occur in $y \in z$".

In the induction step that follows, we shall implicitly use the lemma

(L19) $$\neg\forall x x = y \to \forall z \neg\forall x x = y$$

which in effect states that a distinctor behaves as if no variable were free in it (including y, as can be seen by substituting y for z in L19). According to the Deduction Theorem for predicate calculus (which can be proved for S2 in a manner analogous to that in [4, p. 77], with the slight complication that applications of the substitution rule inside a deduction may affect the assumptions; cf. [8, p. 140]), Lemma L19 allows us to use distinctors as assumptions or hypotheses in a deduction, without having to worry about any undesirable side effects that may result from using the generalization rule inside of a deduction. In what follows we shall assume that all distinctors have been temporarily removed and placed into an assumption list.

If the F in $F \to \forall u F$ is not an atomic wff then $F \to \forall u F$ must have one of the three forms $(G \to H) \to \forall u(G \to H)$, $\neg G \to \forall u \neg G$, or $\forall v G \to \forall u \forall v G$ by

the definition of a wff. As our induction hypothesis, we assume that we have proven the shorter cases of C5, i.e. $G \to \forall uG$ and $H \to \forall uH$ (in the case of $\to$). We can prove $F \to \forall uF$ by first applying the generalization rule as needed to the shorter cases then detaching from a substitution instance of the lemma of the following three that corresponds to the form of $F \to \forall uF$.

(L20) $\forall x(P \to \forall xP) \to ((Q \to \forall xQ) \to ((P \to Q) \to \forall x(P \to Q)))$

(L21) $\forall x(P \to \forall xP) \to (\neg P \to \forall x\neg P)$

(L22) $\forall y(P \to \forall xP) \to (\forall yP \to \forall x\forall yP)$

There will be one assumption of the form $\neg\forall uu = v$ for each variable v in F required to be distinct from u. We use the Deduction Theorem to reattach a conjunction of these assumptions as an antecedent G to the formula $F \to \forall uF$, and we interpret G as the informal phrase "where u is not among the set of variables occurring in F". The antecedent G is discarded when mapping $G \to (F \to \forall uF)$ to system S3 by the method of distinctor elimination. (This completes the proof of axiom schema C5. This proof is analogous to one given by Lemmas 22 through 25 in [13].)

Finally, the rules of S3 follow from the rules of S2 because they are the identical.

9 Further Remarks

Remark 9.1 The theorems of system S2 may be viewed as metatheorems of S3 (i.e. traditional predicate calculus), where the individual variables of S2 range over the variables of S3 and the propositional variables of S2 range over wffs of S3. Three correspondences are useful in this context. First, as described previously, an antecedent in the form of a distinctor $\neg\forall uu = v$ can be interpreted as the requirement that u and v be distinct. Second, if a propositional variable P is prefixed with $\forall u$ throughout a formula, we can interpret $\forall uP$ as a formula metavariable F ranging over wffs in which u is not free. Third, if we interpret a propositional variable P as F, the formula $(u = v \to P) \wedge \exists u(u = v \wedge P)$ can be interpreted as $F(u|v)$, i.e. as the proper substitution of v for u in F; u and v need not be distinct. Note that $F(u|u) \leftrightarrow F$.

Remark 9.2 S2 (or S1) is not complete in the sense that some propositionless formulas corresponding to metatheorems of S3 are not provable in S2 because any dummy variable used in a proof must appear in the result of the proof. In particular, for a given theorem of S3, we cannot know how it may be represented in S2 until we have a proof and thus obtain an upper bound for the number of dummy variables required. To achieve completeness in this sense we can add to S2 the inference rule of *quantifier elimination*: if, in a theorem of S2, a variable u occurs as the first argument of one or more $\forall$ connectives but nowhere else, delete all occurrences of $\forall u$ from the theorem. (An analogous rule

that preserves finite axiomatization can be added to S1.) To use this rule we avoid any reference to B9 and B14 (whose quantifiers are the source of dummy variables) in the construction of C5 in Section 8 and instead use the required instance of C5 (prefixed with quantifiers to bind its variables) as an assumption for the proof. Applied to the result of the proof, the rule of quantifier elimination will reduce these assumptions to (quantified) tautologies that can be discarded.

Remark 9.3 The following result for S2 is useful for eliminating unnecessary distinctors. (It will not eliminate distinctors containing dummy variables.) The proof, whose details we omit, appeals to axiom B11 in particular to construct a proof of $\forall uu = v \rightarrow (F(u,u) \rightarrow F(u,v))$ by induction on the number of connectives in $F(u,v)$.

Theorem 9.4 (Distinctor Reduction Theorem) *Let $F(u,v)$ be any formula that may contain variables u and v, free or bound in any combination, as well as any other variables. Suppose we have proofs of two theorems of the forms: (1) $\neg\forall uu = v \rightarrow F(u,v)$ and (2) $F(u,u)$ where* all *occurrences of v (both free and bound) in $F(u,v)$ are replaced with u. Then $F(u,v)$ is a theorem.*

Remark 9.5 An open question is whether in S2 we can prove

(B15) $$\neg\forall xx = y \rightarrow (x = y \rightarrow (P \rightarrow \forall x(x = y \rightarrow P))).$$

Remark 9.6 We define a *simple metatheorem* as any metatheorem of S3 consisting only of connectives, individual metavariables, and wff metavariables (with no arguments i.e. no explicit substitutions) and possibly accompanied by a set of variable restrictions of the two forms "where u and v are distinct variables" and "where u is not among the set of variables occurring in F." For example, all axiom schemata of S3 except C7 are simple metatheorems. We define a system as *metalogically complete* if all of its simple metatheorems can be proved with *simple metalogic* as follows: each step in the proof must be a simple metatheorem, and the inference metalogic consists only of the two rules of S3, together with the obvious two substitution rules (for individual and wff metavariables) to produce new simple metatheorems from existing ones provided that variable restrictions are not violated. At each step, the set of variable restrictions is adjusted for any substitutions so that each restriction has only two metavariables, and restrictions involving metavariables not contained in the step are dropped. We do not formalize these notions here but it should be clear how to do so, e.g. as in [17].

A metalogically complete system can be advantageous in a machine-oriented application and perhaps in studies of logic since theorem schemata of traditional predicate calculus, not just specific theorems, can be proved directly with simple metalogic.

We define system S3′ as follows. Rewrite B1 through B15 in the formalization of S3 by replacing P with F, x with u, etc. and call them C1′ through C15′. Let S3′ consist of the language and rules of S3 along with axiom schemata C5,

C1′ through C13′, C15′, and

(C16′) $\forall u u = v \to (F \to \forall u F)$, where u and v are distinct variables.

All of the axiom schemata of S3′ are simple metatheorems. (C14′ is omitted from S3′ because it can be proved from the others using only simple metalogic.)

It is not hard to show that S3′ is logically equivalent to S3 and thus has the same set of simple metatheorems; the reasoning is similar to that in Section 8. In addition, S3′ has the following stronger property.

Theorem 9.7 (Metalogical Completeness Theorem) *System* S3′ *is metalogically complete.*

Proof: We describe the main ideas of the proof but leave some details to the reader. We let s and t (as well as u and v) represent individual metavariables and use $F(u|v)$ to abbreviate $(u = v \to F) \wedge \exists u(u = v \wedge F)$. Let H be the simple metatheorem we wish to prove. Let $u_1, \ldots, u_n$ be the distinct individual metavariables in H. We associate with each wff metavariable F_i in H an m-ary predicate P_i where $m = n$ and whose jth argument corresponds to individual metavariable u_j in H, except that we reduce the arity of P_i by one and remove its jth argument for each variable restriction "where u_j does not occur in F_i". We temporarily extend S3′ with these predicates and add equality axioms for them (similar to schema C7 of S3). Let H_e be the formula of extended S3′ obtained by rewriting H with individual metavariables replaced with actual variables $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of S3′ and each wff metavariable F_i replaced with its corresponding extended m-ary predicate $P_i \mathbf{x}_{i_1} \cdots \mathbf{x}_{i_m}$ $(1 \le i_1 < \cdots < i_m \le n)$. Since H_e is an instance of metatheorem H and hence an actual theorem, it has an ordinary proof in extended S3′.

To construct a proof for H in the original S3′, using only simple metalogic, we mimic the proof of H_e. In the proof, in place of variables $\mathbf{x}_1, \ldots, \mathbf{x}_{n+d}$ (where d is the number of distinct dummy variables in the proof) outside of extended predicates we use individual metavariables $u_1, \ldots, u_{n+d}$. In place of an extended predicate occurrence $P_i \mathbf{x}_{j_1} \cdots \mathbf{x}_{j_m}$ $(1 \le j_k \le n + d)$ we use the wff $F_i(u_{i_1}|v_1) \cdots (u_{i_m}|v_m)(v_1|u_{j_1}) \cdots (v_n|u_{j_m})$, where $v_1, \ldots, v_m$ are dummy variables distinct from each other and from $u_1, \ldots, u_{n+d}$ and that do not occur in F_i. We recover F_i from $F_i(u_{i_1}|v_1) \cdots (u_{i_m}|v_m)(v_1|u_{i_1}) \cdots (v_m|u_{i_m})$ at the end of the proof using substitution laws $F(u|v)(s|t) \leftrightarrow F(s|t)(u|v)$ (where u and s are distinct, u and t are distinct, and s and v are distinct) and $F(u|v)(v|u) \leftrightarrow F$ (where v does not occur in F), both provable in S3′ with simple metalogic. (C15′ seems to be needed for these proofs. Since we omit the proofs of these and a few other simple metatheorems, the reader may just regard them as additional, though redundant, axioms of S3′ for the purpose of understanding the main proof.)

In mimicking the proof, all applications of axioms and rules are analogous except for three cases. (1) Whenever an equality axiom for an extended predicate is referenced, we use instead the metatheorem $u = v \to (F(s|u) \to F(s|v))$

(provable in S3′ with simple metalogic; C15′ seems to be needed), manipulating F with the first substitution law above before and after this metatheorem is applied. (2) Whenever schema C5 is referenced, we instead construct with simple metalogic an appropriate metatheorem of the form $F \to \forall u F$ (in a manner similar to the proof of C5 in Section 8, possibly using C5 itself and also making use of the metatheorem $\forall u F \to \forall u \forall u F$ as needed). We place in an assumption list any distinctors that arise. We use C16′ to eliminate a distinctor whenever H has a restriction of the form "where u and v are distinct" and whenever a (distinct) dummy variable is introduced. (3) Whenever schema C16′ is referenced but the variables u and v are not required to be distinct in H, we use instead the tautology $\neg\forall u u = v \to (\forall u u = v \to (F \to \forall u F))$ and place its antecedent in the assumption list of distinctors.

To eliminate any remaining distinctors not part of H, we apply the Distinctor Reduction Theorem (restated in the language of S3′), repeating the entire procedure above to obtain a proof of its special case $F(u, u)$. □

10 ZF Set Theory

Zermelo-Fraenkel (ZF) set theory is not a finitely axiomatizable extension of traditional predicate calculus because the Axiom Schema of Replacement requires a formula metavariable F ranging over certain wffs [3, p. 83]. ("No finite number of axioms of ZF imply all the axioms of ZF.") In the formalization of system S2, however, we can represent a schema containing a wff metavariable F by a formula containing a propositional variable P, allowing the Axiom Schema of Replacement to become a particular axiom. At the primitive level of system S1, this propositional variable is indistinguishable from the other variables but behaves as if it were a formula metavariable when subjected to the axioms and rules of S1. We can always write a formula of S3 in the formalization of S2 by anteceding it with a conjunction of distinctors; in particular, we can state Replacement as

$$(\neg\forall x x = y \wedge \neg\forall x x = z \wedge \neg\forall y y = z) \to \\ (\forall x \exists z \forall y (\forall z P \to y = z) \to \exists x \forall y (y \in x \leftrightarrow \exists x (\forall z P \wedge x \in z))).$$

Boyer et. al. [1] write, "since [ZF] cannot be finitely axiomatized, it cannot be input to a resolution-based theorem prover" and base theirs (see also [19]) on NBG, which is finitely axiomatized (in the customary sense). While ZF cannot be directly input to such a prover, using the formalization of S1 it can be indirectly input via the provability predicate method described in the Introduction. Whether this offers practical advantages is not known.

Appendix: Proofs of the Lemmas in This Paper

We can make use of the fact that Rules **D** and **G** have unique results to communicate, unambiguously, complete formal proofs in system S1 in a compact manner. We denote axioms A1 through A9 by the characters `1` through `9` and A10 through A14 by the characters `a` through `e`. A character string is constructed as follows. `D`qp represents the result of Rule **D** when the theorem expressed by string p is detached from the theorem expressed by string q. `G`p is represents the result of Rule **G** applied to the theorem expressed by string p. We call a proof expressed in this notation a *proof string*.

As an example of how a proof string is constructed, consider a proof of $\forall x P \rightarrow \forall x \forall x P$. For readability the notation of system S2 is used to display the subtheorems, but the proof is implicitly in system S1 and uses Rules **D** and **G** of system S1.

Step	Subtheorem	Reason	Proof string
a.	$(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$	Axiom A2	`2`
b.	$P \rightarrow (Q \rightarrow P)$	Axiom A1	`1`
c.	$(P \rightarrow Q) \rightarrow (P \rightarrow P)$	Rule **D**,a,b	`D21`
d.	$P \rightarrow (Q \rightarrow P)$	Axiom A1	`1`
e.	$P \rightarrow P$	Rule **D**,c,d	`DD211`
f.	$\forall x (P \rightarrow P)$	Rule **G**,e	`GDD211`
g.	$\forall x (\forall x P \rightarrow Q) \rightarrow (\forall x P \rightarrow \forall x Q)$	Axiom A4	`4`
h.	$\forall x P \rightarrow \forall x \forall x P$	Rule **D**,g,f	`D4GDD211`

Thus the proof string for this proof is `D4GDD211`. (The reader may wish to verify that a shorter proof string[2] for this theorem is `D4GD4G5`.)

A proof string, then, is essentially a list of the steps in a formal proof expressed in a Polish prefix notation. A simple computer program (incorporating algorithms **D** and **G**) can scan the proof string in reverse order and reconstruct the formal proof.

In general, proof strings cannot represent the shortest possible proofs in system S1 because subtheorems used more than once must have their proof strings repeated. However, the proof strings below are short enough so that we don't bother to introduce notation that permits references to repeated subtheorems or other lemmas. We use the character `S` to abbreviate `DD2D1` (representing a syllogism inference).

Because they are used to prove the Substitution Theorem, Lemmas L1 through L11 are the exact theorems proved by their proof strings. The remaining lemmas may be substitution instances of the theorems proved by their proof strings and in such cases we implicitly apply the Substitution Theorem.

(L1) *Proof:* `DDD21DD22D211`

[2]Found by the OTTER theorem prover [20].

(L2) *Proof:* DD2SSD2SD2D1DD2S3D2S311SDD2S21D1S3D2D1D3DD2S3S311S211D2D1SSD2-S8DD28D3DD2S31SaSD4GSDDD22D219571S5SDD21bD4GSD3DD2S31SaSD4GSDDD22D219575-D1S8S5SDD21bD4GSD3DD2S31SaSD4GSDDD22D219575

(L3) *Proof:* DD2SSD2SD2D1DD2S3D2S311SDD2S21D1S3D2D1D3DD2S3S311S211D2D1SdS5-SDD21bD4GSD3DD2S31SaSD4GSDDD22D219575D1ScS5SDD21bD4GSD3DD2S31SaSD4GSDDD2-2D219575

(L4) *Proof:* D2D1S3SDDD21DD2D21D1SSDDD21DD2D21D1DD211331D211

(L5) *Proof:* SD2SSSDD22D11S12SD2D1SD2SD2D1DD2S3D2S311SDD2S21D1S3D2D1D3DD2S-3S311S211S21D2D1S21SDD2S21D11S2D2D1S1SD2S211

(L6) *Proof:* DD2SSSDD22D11S12SD2D1SD2SD2D1DD2S3D2S311SDD2S21D1S3D2D1D3DD2S-3S311S211S21D1S4D4GSSDD2S21D15D1DD2115D2D1SSSD2D1D4G5D21DDDD21DD22D2115D-D2S21D15

(L7) *Proof:* DD2S21D1S3D2D1D3DD2S3S311

(L8) *Proof:* DD2S21D1SS3D2D1D3DD2S3S311SSDD22D11S12D2DDD21S3D21D1S3D2D1D3D-D2S3S311

(L9) *Proof:* DD2S21D1DD21D2S31

(L10) *Proof:* DD2S21D1DD21SD2D1DD21D2S31S3D21

(L11) *Proof:* SDD2SD2211DDD21DD2S21S21D1DD211

(L12) *Proof:* D3SD3DD2S3S311SSD4GSDD28D3DD2S31SaSD4GSDDD22D219575DD2bD4G5D-D2S3S311

(L13) *Proof:* SD2S211

(L14) *Proof:* DD2S3D2S311

(L15) *Proof:* SD2S311

(L16) *Proof:* S4D4GSDD2S21D155

(L17) *Proof:* SD2S8DD28D3DD2S31SaSD4GSDDD22D219571

(L18) *Proof:* DaGS7D3DD2S3S311

(L19) *Proof:* SD4GSD3SD3DD2S3S311SS6D4GDD3DD2S31SSSDD2S21D1DD2bD4G5bSD4GSD-D28D3DD2S31SaSD4GSDDD22D219575DD2bD4G5D3SD3DD2S3S311DD3DD2S31SS1SbSD4GSD-D28D3DD2S31SaSD4GSDDD22D219575DD2bD4G5D3SD3DD2S3S311SD2D1DD2S21D15911DD2-S3S31157

(L20) *Proof:* SSD2SSDD22D11S12SD2D1SDD2S21D1S3D2D1D3DD2S3S311SD2D1DD2S3D2S-311S21SS21D2D1D4GS151SD2SS4D4GSSDD2S21D15SS21S31571

(L21) *Proof:* SDD2S21D17S4D4GSSDD2S21D15SS3SDD22D2S3S311D2D1D3DD2S3S3115

(L22) *Proof:* SD2D16S4D4GSDD2S21D155

References

[1] Boyer, Robert, Ewing Lusk, William McCune, Ross Overbeek, Mark Stickel, and Lawrence Wos, "Set theory in first order logic: Clauses for Gödel's axioms," *Journal of Automated Reasoning*, vol. 2 (1986), pp. 287–327.

[2] Church, Alonzo, *Introduction to Mathematical Logic*, Volume 1, Princeton University Press, Princeton, N. J., 1956.

[3] Cohen, Paul J., *Set Theory and the Continuum Hypothesis*, W. A. Benjamin, Inc., Reading, Mass., 1966.

[4] Hamilton, Alan G., *Logic for Mathematicians*, Cambridge University Press, Cambridge, 1988.

[5] Hindley, J. Roger and David Meredith, "Principal type-schemes and condensed detachment," *The Journal of Symbolic Logic*, vol. 55 (1990), pp. 90–105.

[6] Kalish, Donald and Richard Montague, "On Tarski's formalization of predicate logic with identity," *Archiv für Mathematische Logik und Grundlagenforschung*, vol. 7 (1965), pp. 81–101.

[7] Kalman, J. A., "Condensed detachment as a rule of inference," *Studia Logica*, vol. 42 No. 4 (1983), pp. 443–451.

[8] Kleene, Stephen Cole, *Introduction to Metamathematics*, D. Van Nostrand Company, Inc., Princeton (1952).

[9] Megill, Norman D. and Martin W. Bunder, "Weaker D-complete logics," The University of Wollongong Department of Mathematics Preprint Series no. 15/94. Submitted.

[10] Mendelson, Elliott, *Introduction to Mathematical Logic*, second edition, D. Van Nostrand Company, Inc., New York (1979).

[11] Meredith, David, "In memoriam Carew Arthur Meredith (1904–1976)," *Notre Dame Journal of Formal Logic*, vol. XVIII (1977), pp. 513–516.

[12] Mints, Grigori and Tanel Tammet, "Condensed detachment is complete for relevance logic: A computer-aided proof," *Journal of Automated Reasoning*, vol. 7 (1991), pp. 587–596.

[13] Monk, J. Donald, "Substitutionless predicate logic with identity," *Archiv für Mathematische Logik und Grundlagenforschung*, vol. 7 (1965), pp. 103–121.

[14] Nemeti, I., "Algebraizations of quantifier logics, an overview," version 11.4, preprint, Mathematical Institute, Budapest, 1994. A shortened version without proofs appeared in *Studia Logica*, vol. 50 (1991), pp. 485–569.

[15] Peterson, Jeremy George, "An automatic theorem prover for substitution and detachment systems," *Notre Dame Journal of Formal Logic*, vol. XIX (1978), pp. 119–122.

[16] Robinson, J. A., "A machine-oriented logic based on the resolution principle," *Journal of the Association for Computing Machinery*, vol. 12 (1965), pp. 23–41.

[17] Tarski, Alfred, "A simplified formalization of predicate logic with identity," *Archiv für Mathematische Logik und Grundlagenforschung*, vol. 7 (1965), pp. 61–79.

[18] Tarski, Alfred and Steven Givant, *A Formalization of Set Theory Without Variables*, American Mathematical Society Colloquium Publications, vol. 41, American Mathematical Society, Providence, R. I., 1987.

[19] Wos, L., *Automated Reasoning: 33 Basic Research Problems*, Prentice-Hall, Englewood Cliffs, N. J., 1987

[20] Wos, Larry, Ross Overbeek, Ewing Lusk and Jim Boyle, *Automated Reasoning: Introduction and Applications*, second edition, McGraw-Hill, Inc., New York, 1992.

[21] Wos, L. T., and G. A. Robinson, "Maximal models and refutation completeness: Semidecision procedures in automated theorem proving," in William W. Boone, Frank Benjamin Cannonito, and Roger C. Lyndon, editors, *Word Problems: Decision Problems and the Burnside Problem in Group Theory*, pp. 609–639, North-Holland, Amsterdam, 1973, Studies in Logic and the Foundations of Mathematics, vol. 71.

[22] Zeman, J. J., *Modal Logic*, Oxford University Press, Oxford, 1973.