Adding the Everywhere Operator to Propositional Logic

DAVID GRIES and FRED B. SCHNEIDER, *Computer Science, Cornell University, Ithaca, NY 14853, USA.*
E-mail: gries@cs.cornell.edu

Abstract

Sound and complete modal propositional logic C is presented, in which $\Box P$ has the interpretation 'P is *true* in all states'. This interpretation is already known as the Carnapian extension of S5. The new axiomatization for C provides two insights. First, introducing an inference rule *textual substitution* allows integration of the propositional and modal parts of the logic in a way that gives a more practical system for writing formal proofs. Second, the two following approaches to axiomatizing a logic are shown to be not equivalent: (i) give axiom schemes that denote an infinite number of axioms and (ii) write a finite number of axioms in terms of propositional variables and introduce a substitution inference rule.

Keywords: Propositional logic, calculational logic, everywhere operator.

1 Introduction

Logic gives a syntactic way to derive or certify truths that can be expressed in the language of the logic. The expressiveness of the language impacts the logic's utility—the more expressive the language, the more useful the logic (at least if the intended use is to prove theorems, as opposed to, say, studying logics). We wish to calculate with logic, much as one calculates in algebra to derive equations.

Proofs in calculational logic are centred not on inference rule *modus ponens* but on *substitution of equals for equals*. Proofs take the form of sequences of substitutions—see p. 124 for an example. This form of proof enjoys widespread use in mathematics and in applications of math to engineering and science, but it has rarely been used in logic.

The three inference rules of calculational propositional logic, given in Table 1, are designed to support a calculational style of proof. This logic is equivalent to traditional propositional logic [11].

Since the late 1970s, researchers in the field of formal development of programs have made heavy use of first-order predicate logic in writing specifications, program derivations, and program proofs. For them, axiomatizations of calculational logics—not natural deduction systems, tableau methods, etc.—have been the logics of choice. These people write formal proofs as a matter of course in their work.

The growing interest in calculational logic can be seen in the following.

- A special issue of *Information Processing Letters* (53, 3, 10 February 1995) was devoted to articles that used or discussed calculational logics, including papers on reasoning about sequential phenomena, fixed-point calculus, constructing the Galois adjoint, mapping a functional parallel program onto a hypercube, constructing algorithms, and games and winning strategies.
- A discrete math text [8], which is based on calculational logic, aims expressly to impart a

TABLE 1. Inference rules of calculational propositional logic **E**

Leibniz:[1]	$\vdash \alpha \equiv \beta$	$\longrightarrow \vdash \gamma^v_\alpha \equiv \gamma^v_\beta$
Transitivity:	$\vdash \alpha \equiv \beta, \vdash \beta \equiv \gamma$	$\longrightarrow \vdash \alpha \equiv \gamma$
Equanimity:	$\vdash \alpha, \vdash \alpha \equiv \beta$	$\longrightarrow \vdash \beta$

(Here, γ^v_α denotes a copy of formula γ in which each occurrence of variable v is replaced by formula α.)

skill in formal manipulation and to have students apply that skill to other topics typically taught in discrete math—e.g. set theory, a theory of integers, induction, and solving recurrence relations. Students seem much happier with this approach than with the traditional one, where useful applications of formal logic are rarely discussed. In view of our experience with the text, we believe that the approach can help develop excitement for further study of logic. Our views on the approach are discussed in [9, 10].

We find it useful for $\Box P$ (read 'everywhere P') to be a formula of the logic and have the meaning 'P is *true* in all states'. (The everywhere operator is discussed in [5].) When a propositional logic extended with the *everywhere operator* $\Box$ is further extended to predicate logic and then to other theories, the logic can be used for proving theorems that could otherwise be handled only at the metalevel, and most likely informally. For example, the statement

$$P \text{ is valid iff } \forall x.P \text{ is valid} \tag{1.1}$$

is formalized in our logic as $\Box P \equiv \Box(\forall x.P)$. In contrast, formalizing (1.1) as the two inference rules

$$\vdash P \longrightarrow \vdash \forall x.P \qquad \text{and} \qquad \vdash \forall x.P \longrightarrow \vdash P$$

demotes it to a meta-logical notion. This is awkward because the equivalence of P and $\forall x.P$ is not expressible by a formula of the logic and thus is not directly available for use in calculational reasoning.

As another example, the following fact about set theory,[1]

$$\{x \mid Q\} = \{x \mid R\} \text{ is valid iff } Q \equiv R \text{ is valid,}$$

is formalized using $\Box$ as

$$\Box(\{x \mid Q\} = \{x \mid R\}) \;\equiv\; \Box(Q \equiv R),$$

but it cannot be formalized as a formula without something like $\Box$.

Finally, the everywhere operator can be used to formalize $\{P\}\ S\ \{Q\}$, the Hoare triple, with meaning 'execution of statement S begun with P *true* is guaranteed to terminate with Q *true*' [14]. Using weakest-precondition predicate transformer wp [4], we define:

$$\{P\}\ S\ \{Q\}: \ \Box(P \Rightarrow wp(S, Q)).$$

[1] We use $\equiv$ for equality over the Booleans and $=$ for equality over any type.

TABLE 2. Table of abbreviations

$\alpha \wedge \beta : \neg(\neg\alpha \vee \neg\beta)$	$true : p \equiv p$
$\alpha \Rightarrow \beta : \neg\alpha \vee \beta$	$false : \neg true$
$\alpha \equiv \beta : (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$	$\diamond\alpha : \neg\Box\neg\alpha$

Modal logic[2] S5 includes $\Box P$ among its formulas. As is well known, S5 is not complete with respect to the Carnapian model C, which consists of all states (total functions from the set of all propositional variables to $\{\mathbf{t}, \mathbf{f}\}$, with the conventional definition of evaluation), where every state is accessible from every other state. For example, the formula $\neg\Box p$ for p a propositional variable is valid w.r.t. this model, but it is not a theorem of S5.

Several sound and complete axiomatizations for model C are known [2, 3, 12, 15, 17], dating from as early as 1973—see Gottlob's survey [7]. In Section 3, we give a new axiomatization for model C, compare it with previous ones, and argue why we believe the new axiomatization is more suitable for actually writing formal proofs.

The axiomatization of Section 3 uses an infinite number of axioms, specified by a finite set of axiom schemes. In Section 4 we present an axiomatization with a finite number of axioms. We show that such a finite axiomatization cannot be obtained simply by replacing the metavariables of the axiom schemes of the infinite axiomatization (of Section 3) by propositional variables and adding inference rule Uniform Substitution. This, then, demonstrates that the two approaches to axiomatizing a logic are not necessarily equivalent.

2 Preliminaries

Let VP be a set of propositional variables. We use lower-case letters $p, q, r, \ldots$ for elements of VP. A *formula* of S5 has one of the following forms (p is any variable in VP, and metavariables α, β stand for formulas).

$$p \qquad (\neg\alpha) \qquad (\alpha \vee \beta) \qquad (\Box\alpha) \tag{2.1}$$

In addition, $(\alpha \wedge \beta)$, $(\alpha \Rightarrow \beta)$, $(\alpha \equiv \beta)$, $\diamond\alpha$, $true$, and $false$ are abbreviations of certain formulas, as shown in Table 2. (Operator $\diamond$ is read as 'possibly' or 'somewhere'.) Precedences eliminate the need for some parentheses; prefix operators $\neg$, $\Box$, and $\diamond$ bind tightest, then $\vee$ and $\wedge$, then $\Rightarrow$, and finally $\equiv$.

A formula of S5 that contains neither $\Box$ nor $\diamond$ is called a *propositional formula*.

A *model* is a triple $\langle W, R, V\rangle$ in which:

- W is a nonempty set of *worlds*.
- R is an *accessibility relation*, a binary relation over W: $w\ R\ u$ signifies that *world u is accessible from world w*.
- $V(\alpha, w)$, for α a formula and w a world in W, is a *value assignment* that satisfies the following properties:

[2] See Hughes and Cresswell [16] for an introduction to modal logic.

TABLE 3. Schematic PM, S5, and C

PM:	**Modus Ponens:** $\vdash \alpha, \vdash (\alpha \Rightarrow \beta) \longrightarrow \vdash \beta$
	Axiom scheme A1: $\alpha \vee \alpha \Rightarrow \alpha$
	Axiom scheme A2: $\alpha \Rightarrow \alpha \vee \beta$
	Axiom scheme A3: $\alpha \vee \beta \Rightarrow \beta \vee \alpha$
	Axiom scheme A4: $(\beta \Rightarrow \gamma) \Rightarrow (\alpha \vee \beta \Rightarrow \alpha \vee \gamma)$
S5:	**Necessitation:** $\vdash \alpha \longrightarrow \vdash \Box\alpha$
	Axiom scheme $\Box$-Instantiation: $\Box\alpha \Rightarrow \alpha$
	Axiom scheme Monotonicity: $\Box(\alpha \Rightarrow \beta) \Rightarrow (\Box\alpha \Rightarrow \Box\beta)$
	Axiom scheme Necessarily Possible: $\Diamond\alpha \Rightarrow \Box\Diamond\alpha$
C:	**Textual Substitution:** $\vdash \gamma \longrightarrow \vdash \gamma[v := \beta]$

$$
\begin{array}{lll}
V(p,w) & \text{is} & \text{either } \mathbf{t} \text{ or } \mathbf{f} \text{ (for } p \text{ a variable in } VP), \\
V(\neg\alpha,w) & = & \mathbf{if}\ V(\alpha,w)=\mathbf{t}\ \mathbf{then}\ \mathbf{f}\ \mathbf{else}\ \mathbf{t}, \\
V(\alpha \vee \beta,w) & = & \mathbf{if}\ V(\alpha,w)=\mathbf{t}\ \mathbf{then}\ \mathbf{t}\ \mathbf{else}\ V(\beta,w), \\
V(\Box\alpha,w) & = & \mathbf{if}\ V(\alpha,u)=\mathbf{t} \text{ for all worlds } u \text{ such that } w\ R\ u\ \mathbf{then}\ \mathbf{t}\ \mathbf{else}\ \mathbf{f}.
\end{array}
$$

An S5-*model* is a model $\langle W, R, V\rangle$ in which R is an equivalence relation—reflexive, transitive, and symmetric. An S5-formula α is S5-*valid*, written $\models_{S5} \alpha$, iff for every S5-model $\langle W, R, V\rangle$ and every w in W, $V(\alpha, w) = \mathbf{t}$.

The first part of Table 3 is a schematic presentation of propositional logic PM. PM consists of one inference-rule scheme (Modus Ponens) and four axiom schemes. The inference-rule scheme denotes the infinite set of inference rules constructed by replacing metavariables α and β by formulas. (Similarly for the axiom schemes.) In PM (as in all the logics in this paper), a *theorem* is either an axiom or the conclusion of an inference rule whose premisses are theorems. We use the notation $\vdash_L \alpha$ for 'α is a theorem of logic L'. ($\vdash_L$ applies to the longest formula that follows it.)

The second part of Table 3 extends propositional logic PM to modal logic S5, by adding one inference-rule scheme and three axiom schemes. S5 is sound and complete with respect to S5-validity [16].

3 Logic C

The Carnapian model C is the conventional set of all states (total functions from the set of all propositional variables to $\{\mathbf{t}, \mathbf{f}\}$), together with the universal accessibility relation (i.e. all states are accessible from each state) and the conventional definition of evaluation. In this model, $\Box\alpha$ has the interpretation 'α is true in all states'. We now define this model formally.

Let $\widehat{W}$ be the set of all total functions $w : VP \to \{\mathbf{t}, \mathbf{f}\}$. Let $\widehat{R}$ be the universal relation over $\widehat{W}$, i.e. $w\ \widehat{R}\ u$ holds for all w, u in $\widehat{W}$. Let $\widehat{V}$ be the value assignment defined by $\widehat{V}(p, w) = w.p$.[3] We define $\langle \widehat{W}, \widehat{R}, \widehat{V}\rangle$ to be the (only) C-model and define a formula α to be C-valid iff for every w in $\widehat{W}$, $\widehat{V}(\alpha, w) = \mathbf{t}$. The C-model is also an S5-model, since $\widehat{R}$ is an equivalence relation. Consequently, $\models_{S5} \alpha$ implies $\models_C \alpha$.

[3] For f a one-parameter function, $f.x$ denotes the application of f to argument x.

Logic S5 is sound but not complete with respect to C-validity. To see this, consider the formula $\neg\Box p$. It is not S5-valid, since it evaluates to $\mathbf{f}$ in the S5-model $(\{w\}, I, V)$, where $V(p, w) = \mathbf{t}$ and I is the identity relation. Since S5 is sound, $\neg\Box p$ is not a theorem of S5. However, $\models_C \neg\Box p$ holds, since $\Box p$ evaluates to $\mathbf{f}$—there is a world w in $\widehat{W}$ that satisfies $\widehat{V}(p, w) = \mathbf{f}$.

We will need the notion of *textual substitution* $\gamma[v := \beta]$ where v is a propositional variable and γ and β are formulas. The recursive definition given below treats occurrences of variables in $\Box\alpha$ as if they were bound.

$$\begin{array}{lll}
v[v := \beta] & = \beta & \\
u[v := \beta] & = u & \text{(for variable } u \text{ different from } v) \\
(\neg\gamma)[v := \beta] & = \neg(\gamma[v := \beta]) & \\
(\delta \circ \gamma)[v := \beta] & = \delta[v := \beta] \circ \gamma[v := \beta] & \text{(for binary connective } \circ) \\
(\Box\gamma)[v := \beta] & = \Box\gamma. &
\end{array}$$

We allow simultaneous textual substitution, by letting v and β be lists of distinct propositional variables and formulas, respectively. The formal definition is left to the reader.

The third part of Table 3 extends logic S5 with inference-rule scheme Textual Substitution to yield logic C. It is easy to show that Textual Substitution preserves C-validity.

Textual Substitution and all the inference rules of S5 preserve C-validity. Also, the axioms of S5 are C-valid. Therefore, logic C is sound with respect to C-validity.

To illustrate logic C, we prove that $\neg\Box p$ is a theorem. We use a calculational style of proof—see [8] or [11]. The first formula is a C-theorem. Since the last formula equals the first, the last is also a C-theorem.

$$\begin{array}{ll}
& (\Box p \Rightarrow p)[p := \mathit{false}] \text{—Textual Substitution in Axiom } \Box\text{-Instantiation} \\
= & \langle \text{Definition of textual substitution for propositional variable } p \rangle \\
& \Box p \Rightarrow \mathit{false} \\
= & \langle \text{Propositional theorem } Q \Rightarrow \mathit{false} \equiv \neg Q \rangle \\
& \neg\Box p.
\end{array}$$

Proving completeness of logic C *with respect to* C*-validity*

We now prove that logic C is complete with respect to C-validity. Since C is an extension of S5, in this proof, we can use S5-theorems presented in Hughes and Cresswell [16] as C-theorems. Also, we rely on the following lemma, which follows directly from the definition of $\widehat{V}(\Box\alpha, w)$.

LEMMA 3.1
For any formula α, either $\models_C \Box\alpha$ or $\models_C \neg\Box\alpha$.

Hughes and Cresswell define *ordered modal conjunctive normal form* (ordered MCNF). A formula is in ordered MCNF if it has the form $C_1 \wedge \ldots \wedge C_n$ and each C_i has the form

$$\beta \vee \Box\gamma_1 \vee \ldots \vee \Box\gamma_m \vee \Diamond\delta, \tag{3.1}$$

where β, the γ_i, and δ are propositional formulas (i.e. they don't contain $\Box$ or $\Diamond$).

We prove three lemmas, leading up to a proof that $\models_C$ (3.1) implies $\vdash_C$ (3.1).

LEMMA 3.2
For propositional formula β, $\models_C \beta$ implies $\vdash_C \beta$.

PROOF. Suppose $\models_C \beta$. Since β is a propositional formula, and since C contains complete propositional logic PM, $\vdash_C \beta$. ∎

LEMMA 3.3
For propositional formula δ, $\models_C \diamond\delta$ implies $\vdash_C \diamond\delta$.

PROOF. If $\models_C \diamond\delta$, there is a world w such that $\widehat{V}(\delta, w) = \mathbf{t}$. Thus, there is an assignment $[\overline{p} := \overline{c}]$ of constants $\overline{c}$ (each constant being *true* or *false*) to the propositional variables $\overline{p}$ of δ such that $\delta[\overline{p} := \overline{c}]$ evaluates to $\mathbf{t}$ in w. Since $\delta[\overline{p} := \overline{c}]$ contains no variables, it evaluates to $\mathbf{t}$ in all worlds and is valid. By Lemma 3.2, $\vdash_C \delta[\overline{p} := \overline{c}]$.

The following calculational proof shows that $(\Box\neg\delta \Rightarrow \neg\delta)[\overline{p} := \overline{c}]$ is equivalent to $\diamond\delta$. Further, since the first formula is a theorem (it is an instance of axiom $\Box$-Instantiation on which Textual Substitution is performed), the last formula is also a theorem. This establishes $\vdash_C \diamond\delta$.

$$
\begin{array}{cl}
 & (\Box\neg\delta \Rightarrow \neg\delta)[\overline{p} := \overline{c}] \quad \text{—Textual Substitution in } \Box\text{-Instantiation} \\
= & \langle \text{Contrapositive; Double negation} \rangle \\
 & (\delta \Rightarrow \neg\Box\neg\delta)[\overline{p} := \overline{c}] \\
= & \langle \text{Definition of textual substitution} \rangle \\
 & \delta[\overline{p} := \overline{c}] \Rightarrow \neg\Box\neg\delta \\
= & \langle \delta[\overline{p} := \overline{c}] \equiv \mathit{true} \text{ (since } \vdash_C \delta[\overline{p} := \overline{c}]\text{); Abbreviation (see Table 2)} \rangle \\
 & \mathit{true} \Rightarrow \diamond\delta \\
= & \langle \text{Left identity of } \Rightarrow \rangle \\
 & \diamond\delta.
\end{array}
$$

∎

LEMMA 3.4
For propositional formula γ, $\models_C \Box\gamma$ implies $\vdash_C \Box\gamma$.

PROOF. If $\models_C \Box\gamma$, then $\widehat{V}(\gamma, w) = \mathbf{t}$ in all worlds w. Hence, $\models_C \gamma$. By Lemma 3.2, $\vdash_C \gamma$. By inference rule Necessitation, $\vdash_C \Box\gamma$. ∎

THEOREM 3.5
$\models_C$ (3.1) implies $\vdash_C$ (3.1).

PROOF. Suppose $\models_C$ (3.1). By Lemma 3.1, each of $\Box\gamma_i$ and $\diamond\delta$ (i.e. $\neg\Box\neg\delta$) evaluates to $\mathbf{f}$ in all worlds or to $\mathbf{t}$ in all worlds. The proof uses a three-case analysis: $\diamond\delta$ evaluates to $\mathbf{t}$, $\Box\gamma_i$ evaluates to $\mathbf{t}$ for some γ_i, and all $\Box\gamma_i$ and $\diamond\delta$ evaluate to $\mathbf{f}$ in all worlds.

Case $\diamond\delta$ evaluates to t in all worlds. Then $\models_C \diamond\delta$ and, by Lemma 3.3, $\vdash_C \diamond\delta$. Note that $\diamond\delta \Rightarrow$ (3.1) is of the form $P \Rightarrow P \vee Q$, which is a theorem of propositional logic PM, so $\vdash_C \diamond\delta \Rightarrow$ (3.1). By Modus Ponens, $\vdash_C$ (3.1).

Case $\Box\gamma_i$ evaluates to t in all worlds. The proof is similar to the proof of the previous case, using Lemma 3.4 instead of (3.3).

Case the $\Box\gamma_i$ and $\diamond\delta$ evaluate to f in all worlds. Since (3.1), i.e. $\beta \vee \Box\gamma_1 \vee \ldots \vee \Box\gamma_m \vee \diamond\delta$, is C-valid, β evaluates to $\mathbf{t}$ in all worlds, so $\models_C \beta$. The rest of this proof is similar to the proof of the first case, using Lemma 3.2 instead of (3.3). ∎

Hughes and Cresswell [16, p. 117] prove the following theorem.

THEOREM 3.6 (ORDERED MCNF THEOREM)
For any formula α, there exists an ordered MCNF formula $mcnf.\alpha$ such that $\vdash_{S5} \alpha \equiv mcnf.\alpha$.

COROLLARY 3.7
$\vdash_C \alpha \equiv mcnf.\alpha$.

COROLLARY 3.8
$\vdash_C \alpha$ iff $\vdash_C mcnf.\alpha$.

COROLLARY 3.9
$\models_C \alpha$ iff $\models_C mcnf.\alpha$.

Corollary 3.7 holds because C is an extension of S5. Corollary 3.8 follows from Corollary 3.7, the definition of abbreviation $\equiv$, and Modus Ponens. For Corollary 3.9, note that Corollary 3.7 together with the soundness of logic C yields $\models_C \alpha \equiv mcnf.\alpha$ and use properties of $\widehat{V}$.

To prove completeness of logic C, we use the following properties of propositional logic (which is included in logic C) and model C:

$$\vdash_C \alpha \wedge \beta \text{ iff } (\vdash_C \alpha) \text{ and } (\vdash_C \beta), \tag{3.2}$$
$$\models_C \alpha \wedge \beta \text{ iff } (\models_C \alpha) \text{ and } (\models_C \beta). \tag{3.3}$$

THEOREM 3.10
For any formula α, $\models_C \alpha$ implies $\vdash_C \alpha$.

PROOF. $mcnf.\alpha$ has the form $C_1 \wedge \ldots \wedge C_n$ where each C_i has form (3.1). We have,

$$
\begin{array}{ll}
 & \models_C \alpha \\
\text{iff} & \quad \langle \text{Corollary 3.9 —where } mcnf.\alpha \text{ is } C_1 \wedge \ldots \wedge C_n \rangle \\
 & \models_C C_1 \wedge \ldots \wedge C_n \\
\text{iff} & \quad \langle (3.3),\ n-1 \text{ times} \rangle \\
 & (\models_C C_1) \text{ and } \ldots \text{ and } (\models_C C_n) \\
\text{implies} & \quad \langle \text{Monotonicity of and , Theorem 3.5 } (n \text{ times}) \rangle \\
 & (\vdash_C C_1) \text{ and } \ldots \text{ and } (\vdash_C C_n) \\
\text{iff} & \quad \langle (3.2),\ n-1 \text{ times} \rangle \\
 & \vdash_C C_1 \wedge \ldots \wedge C_n \\
\text{iff} & \quad \langle \text{Corollary 3.8 —where } mcnf.\alpha \text{ is } C_1 \wedge \ldots \wedge C_n \rangle \\
 & \vdash_C \alpha.
\end{array}
$$

■

Comparison with earlier complete axiomatizations

Axiomatizations of model C in the literature [2, 3, 12, 15, 17] are all similar in nature to the following one, which we take from [12]. Begin with Schematic S5 (see Table 3). Instead of adding inference rule Textual Substitution, add as axioms all formulas of the form $\diamond\delta$ for δ a satisfiable propositional formula (i.e. a propositional formula that evaluates to **t** in at least one world of model C). Lemma 3.3 now holds trivially, and, since its proof was the only place where inference rule Textual Substitution was used, we can prove completeness of the new logic with respect to C-validity in the same way that we proved completeness of C.

This axiomatization is unsatisfactory to us because it refers to the semantic notion of satisfiability. However, this semantic notion can be eliminated, leading to a complete syntactic axiomatization. A propositional formula is satisfiable iff its disjunctive normal form contains a disjunct that does not contain some literal together with its negation. Hence, to discover whether $\diamond\delta$ (for δ a propositional formula) is a theorem, convert δ to disjunctive normal form and determine whether one of its disjuncts contains a literal and its negation. (Private communications with Rob Goldblatt and Joe Halpern).

The resulting axiomatization is still unsatisfactory to us, because it constricts us to a particular style of proof—to prove a conjectured theorem $\diamond\delta$, δ must be put in disjunctive normal form. For example, to prove $\diamond\delta \lor \diamond\gamma$, we would be forced to prove that one of δ and γ were satisfiable, presumably by reducing δ or γ to disjunctive normal form, rather than simply performing syntactic manipulations to obtain $\diamond\delta \lor \diamond\gamma$, as is our preference. Inference rule Textual Substitution provides an alternative that is more in tune with the way we prove theorems calculationally.

Several interesting properties of operator $\Box$ are presented in a number of recent papers, e.g. [1, 6, 13, 18].

4 C with a finite number of axioms

An axiomatization with a finite number of axioms is often derived from one with axiom schemes by (i) replacing the metavariables in the axiom schemes with propositional variables and (ii) introducing an inference rule to substitute formulas for propositional variables:

$$\vdash \alpha \longrightarrow \vdash \alpha^{v}_{\beta}. \tag{4.1}$$

Here, α is a metavariable, v is a list of propositional variables, and β is a corresponding list of metavariables. The notation α^{v}_{β} is being used to denote a copy of the formula denoted by α in which all occurrences (even those within the scope of $\Box$) of the variables of v are replaced by the formulas denoted by the corresponding variables of β.

This method for removing axiom schemes does not work in the case of Schematic C of Table 3, because (4.1) does not preserve C-validity. For example, $\neg\Box p$ is C-valid (as proven earlier), but $(\neg\Box p)^{p}_{true}$, which is $\neg\Box true$, is not C-valid.

We present a sound axiomatization of model C with a finite number of axioms as follows. Let L be the language of logic C. We extend L to a language L'. Then, we give an axiomatization C$'$ of L'—using a finite number of axioms. Finally, we show that the theorems of C$'$ that are in L are precisely the theorems of logic C.

Let VF be a set of (new) *formula variables*. Upper-case letters $P, Q, R, \ldots$ will denote formula variables. Formulas of language L' are defined as in (2.1), except that a formula variable is also considered to be a formula. For example, $p \lor q$, $P \lor Q$, and $p \lor Q$ are formulas of L'.

A formula of L' is defined to be *concrete* if it does not contain a formula variable. For example, $p \lor q$ is concrete, but $P \lor Q$ and $p \lor Q$ are not concrete. Language L contains exactly the concrete formulas of L'. For a formula α, let $\underline{\alpha}$ denote the concrete formula obtained by replacing every formula variable P of α by the corresponding propositional variable p.

An axiomatization C$'$ of language L' is given in Table 4. Its axioms are those of logic C, except that metavariables have been replaced by formula variables. The inference rules of C$'$ include those of logic C (even to requiring that, in Textual Substitution, γ and β be concrete). Inference rule Uniform Substitution is used only for replacing formula variables: α^{γ}_{β} denotes a copy of the formula denoted by α in which all occurrences (even those within the scope of $\Box$) of the formula variables in list γ are replaced by the formulas denoted by the corresponding variables of β.

TABLE 4. PM′, S5′, and C′

PM′:	**Uniform Substitution:** $\vdash \alpha \longrightarrow \vdash \alpha^{\gamma}_{\beta}$ (γ: list of formula variables)
	Modus Ponens: $\vdash \alpha, \vdash (\alpha \Rightarrow \beta) \longrightarrow \vdash \beta$
	Axiom A1: $P \vee P \Rightarrow P$
	Axiom A2: $P \Rightarrow P \vee Q$
	Axiom A3: $P \vee Q \Rightarrow Q \vee P$
	Axiom A4: $(Q \Rightarrow R) \Rightarrow (P \vee Q \Rightarrow P \vee R)$
S5′:	**Necessitation:** $\vdash \alpha \longrightarrow \vdash \Box\alpha$
	Axiom $\Box$-Instantiation: $\Box P \Rightarrow P$
	Axiom Monotonicity: $\Box(P \Rightarrow Q) \Rightarrow (\Box P \Rightarrow \Box Q)$
	Axiom Necessarily Possible: $\Diamond P \Rightarrow \Box\Diamond P$
C′:	**Textual Substitution:** $\vdash \gamma \longrightarrow \vdash \gamma[v := \beta]$ (γ, β concrete)

One may view logic C′ as simulating metavariables by formula variables. Note that neither Uniform Substitution nor Textual Substitution can be used to derive a non-concrete theorem from a concrete theorem.

We wish to prove that logics C and C′ have the same concrete theorems. To this end, call a Hilbert-style C′ proof *concrete* iff the only non-concrete theorems in it are axioms. This implies that an axiom is used only as the premiss of an instance of Uniform Substitution whose conclusion is concrete. For example, here is a concrete proof of $\neg p \vee (p \vee p) \Rightarrow \neg p \vee p$.

1.	$(Q \Rightarrow R) \Rightarrow (P \vee Q \Rightarrow P \vee R)$	Axiom A4
2.	$(p \vee p \Rightarrow p) \Rightarrow (\neg p \vee (p \vee p) \Rightarrow \neg p \vee p)$	Uniform Substitution
3.	$P \vee P \Rightarrow P$	Axiom A1
4.	$p \vee p \Rightarrow p$	Uniform Substitution
5.	$\neg p \vee (p \vee p) \Rightarrow \neg p \vee p$	Modus Ponens, 4, 2

In a concrete C′ proof, each theorem that is derived using a Uniform Substitution inference is an axiom of C. Thus, the concrete C′ proof can be turned into a C proof simply by deleting each axiom and changing every hint 'Uniform Substitution' to 'Axiom'. For example, the C proof corresponding to the above concrete proof is as follows.

1.	$(p \vee p \Rightarrow p) \Rightarrow (\neg p \vee (p \vee p) \Rightarrow \neg p \vee p)$	Axiom A4
2.	$p \vee p \Rightarrow p$	Axiom A1
3.	$\neg p \vee (p \vee p) \Rightarrow \neg p \vee p$	Modus Ponens, 2, 1

Further, the reverse transformation turns a C proof into a concrete C′ proof. Hence:

THEOREM 4.1
Every theorem of C has a concrete C′ proof, and every concrete theorem of C′ that has a concrete proof is a theorem of C.

We now prove the important

THEOREM 4.2
Every concrete theorem of C′ has a concrete C′ proof.

PROOF. Consider an arbitrary concrete theorem α and a Hilbert-style proof for it. We prove by induction on the length of its proof that there exists a concrete proof of α. Since α is concrete, the proof uses at least one inference rule. Below, we consider the four possibilities for the last inference rule.

Textual Substitution, $\vdash \gamma \quad \longrightarrow \ \vdash \gamma[v := \beta]$. Hence, γ is concrete, and by the induction hypothesis, it has a concrete proof. Since the step $\vdash \gamma \quad \longrightarrow \ \vdash \gamma[v := \beta]$ does not introduce a non-concrete theorem, the result follows in this case.

Necessitation, $\vdash \alpha \quad \longrightarrow \ \vdash \Box\alpha$. Similar to the previous case.

Modus Ponens, $\vdash \gamma,\ \gamma \Rightarrow \alpha \quad \longrightarrow \ \vdash \alpha$. Thus, $\vdash \gamma$ and $\vdash \gamma \Rightarrow \alpha$, so by Uniform Substitution, $\vdash \underline{\gamma}$ and $\vdash \underline{\gamma} \Rightarrow \alpha$ (recall that α is already concrete). By the induction hypothesis, there are concrete proofs of $\underline{\gamma}$ and $\underline{\gamma} \Rightarrow \alpha$. Now use Modus Ponens, $\vdash \underline{\gamma},\ \underline{\gamma} \Rightarrow \alpha \quad \longrightarrow \ \vdash \alpha$, to complete a concrete proof of α.

Uniform Substitution, $\vdash \gamma \quad \longrightarrow \ \vdash \gamma^P_\beta$ for P a list of formula variables. Here, α is γ^P_β. In the left column of Table 5 are the five ways in which the last two steps of the proof could be written. We have abbreviated the names of inference rules by their initials, and we have listed either the premisses or the numbers of lines on which the premisses appear in the proof. In the right column, we give alternative concrete proofs—the formulas on lines with bold-face numbers are concrete formulas for which the inductive hypothesis is assumed, so they have concrete proofs. Since the remaining lines of these proofs contain concrete formulas, the proofs are concrete. ∎

COROLLARY 4.3
For concrete formula α, $\vdash_C \alpha$ iff $\vdash_{C'} \alpha$.

A bit more can be said about formulas of S5 and S5′. Call a formula *abstract* if it contains no propositional variables. Thus, $P \vee P$ is abstract. Consider any formula α that does not contain both a formula variable P and the corresponding propositional variable p. Let $\overline{\alpha}$ denote the abstract formula obtained by changing all (lower-case) propositional variables to (upper-case) formula variables. For example, if α is $p \vee Q$, then $\overline{\alpha}$ is $P \vee Q$ and $\underline{\alpha}$ is $p \vee q$. The proof of the following theorem is left to the reader.

THEOREM 4.4
Let α be a formula that does not contain both a formula variable (e.g. P) and its propositional counterpart (e.g. p). Then the following are all equivalent: $\vdash_{S5} \underline{\alpha}$, $\vdash_{S5'} \overline{\alpha}$, $\vdash_{S5'} \alpha$, $\vdash_{S5'} \underline{\alpha}$, and $\models_{S5} \underline{\alpha}$.

Acknowledgements

We thank Rutger Dijkstra, Rob Goldblatt, Joe Halpern, K. Rustan M. Leino, members of the Eindhoven Tuesday Afternoon Club, and the referees for their extremely helpful comments on drafts of this paper.

The first author was supported in part by NSF grant CDA-9214957 and ARPA/ ONR grant N00014-91-J-4123.

The second author was supported in part by ARPA/RADC grant F30602-96-1-0317, NASA/ARPA grant NAG-2-893,and AFOSR grant F49620-94-1-0198.

TABLE 5. The five possible proofs of γ^P_β

(0) γ (1) γ^P_β	Axiom U.S. (0)	The proof to the left is concrete	
(0) $\delta[v := \sigma]$ (1) $(\delta[v := \sigma])^P_\beta$	T.S. δ U.S. (0)	(0) $\delta[v := \sigma]$ (Textual Substitution requires δ to be concrete, so substituting for P has no effect)	T.S. δ
(0) $\Box\gamma$ (1) $(\Box\gamma)^P_\beta$	N. γ U.S. (0)	(0) γ^P_β (1) $\Box(\gamma^P_\beta)$ (i.e. $(\Box\gamma)^P_\beta$)	U.S. γ N. (0)
(0) γ (1) γ^P_β	M.P. δ, $\delta \Rightarrow \gamma$ U.S. (0)	(0) $\delta^{P,Q}_{\beta,q}$ (1) $(\delta \Rightarrow \gamma)^{P,Q}_{\beta,q}$ (2) $\gamma^{P,Q}_{\beta,q}$	U.S. δ U.S. $\delta \Rightarrow \gamma$ M.P. (0), (1)
(γ^P_β is concrete, so P contains all formula variables in γ. Let Q be a list of formula variables in δ except those in P and let q be a corresponding list of propositional variables. Then γ^P_β is concrete and $\gamma^{P,Q}_{\beta,q}$ is the same as γ^P_β.)			
(0) δ^Q_σ (1) $(\delta^Q_\sigma)^P_\beta$	U.S. δ U.S. (0)	(0) $\delta^{P,Q}_{\beta,\sigma^P_\beta}$	U.S. δ

References

[1] P. Blackburn and E. Spaan. A model perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language and Information*, **2**, 129–169, 1993.

[2] M. J. Carroll. An axiomatization of S13. *Philosophia, Philosophical Quarterly of Israel*, **8**, 381–392, 1978.

[3] N. B. Cocciarella. Logical atomism and modal logic. *Philosophia, Philosophical Quarterly of Israel*, **4**, 41–66, 1974.

[4] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, Englewood Cliffs, NJ, 1976.

[5] E. W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics*. Springer Verlag, New York, 1990.

[6] V. Goranko and S. Passy. Using the universal modality: gains and questions. *Journal of Logic and Computation*, **2**, 5–30, 1992.

[7] G. Gottlob. Survey on a Carnapian extension of S5. Technical report, Institut für Informationssysteme, Technische Universität Wien, 1996. To appear in *Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa*, E. Orlowaska, ed. Kluwer Academic Publishers, Dordrecht.

[8] D. Gries and F. B. Schneider. *A Logical Approach to Discrete Math*. Springer Verlag, New York, 1993.

[9] D. Gries and F. B. Schneider. Teaching math more effectively through calculational proofs. *The Mathematical Monthly*, 691-697, October 1995.

[10] D. Gries and F. B. Schneider. A new approach to teaching discrete mathematics. *Primus V*, **2**, 114–115, 1995.

[11] D. Gries and F. B. Schneider. Equational propositional logic. *Information Processing Letters*, **53**, 145–152, 1995.

[12] J. Y. Halpern and B. Kapron. Zero-one laws for modal logic. *Annals of Pure and Applied Logic*, **69**, 157–193, 1994.

[13] E. Hemaspaandra. The price of universality. *Notre Dame Journal of Formal Logic*, **37**, 174–203, 1996.

[14] C. A. R. Hoare. An axiomatic basis for computer programming. *Commumications of the ACM*, **12**, 576–580, 583, 1969.

[15] H. E. Hendry and M. L. Pokriefka. Carnapian extensions of S5. *Journal of Philosophical Logic*, **14**, 111–128, 1985.

[16] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Mehuen, New York, 1968.

[17] S. K. Thomason. A new representation of S5. *Notre Dame Journal of Formal Logic*, **XIV**, 281–284, 1973.

[18] F. Wolter. Solution to a problem of Goranko and Passy. *Journal of Logic and Computation*, **2**, 5–30, 1992.

Received 11 June 1996