Automatic Construction of Verification Condition Generators From Hoare Logics

Mark Moriconi and Richard L. Schwartz

Computer Science Laboratory
SRI International
Menlo Park, CA 94025

Abstract. We define a method for mechanically constructing verification condition generators from a useful class of Hoare logics. Any verification condition generator constructed by our method is shown to be sound and deduction-complete with respect to the associated Hoare logic. The method has been implemented.

1. Introduction

A *verification condition generator* (VCG), a central component in a program verification system, reduces the question of whether a program is consistent with its specifications to that of whether certain logical formulas are theorems in an underlying theory. VCGs must embody the semantics of the programming language; for the most part, they have been seen as implementations of Hoare-style axiomatic semantics. In the past, all such VCGs have been hand-coded for a specific language, with no formal guarantee that they accurately reflect the axiomatic language definition. The new contributions of this paper are (i) a general method for constructing VCGs mechanically from a useful class of Hoare logics and (ii) a formal basis for the method that provides the needed correspondence between a VCG and the axiomatic definition on which it is based.

Our theoretical results show that any VCG constructed by our method accurately reflects the axiomatic definition of the programming language. In other words, any such VCG is *sound* and *deduction-complete* with respect to the Hoare axiomatization of the language. Of course, it is still necessary to establish the soundness and relative completeness of the axiomatic definition with respect to an operational model [2].

In the process of trying to prove that our method has these properties, we found some subtle limitations of the commonly used implementation strategy for Hoare logics. This led us to identify precise conditions on Hoare logics under which this strategy will produce correct VCGs. Roughly the entire Pascal [5] and Euclid [7] axiomatizations satisfy our constraints, for example. We discuss the practical limitations of this work and propose extensions in the conclusion.

Our method consists of two main steps. An axiomatic definition is first transformed into a normal form from which we then derive a recursively defined VCG. The method has been implemented to form a meta verification condition generator, called MetaVCG. If supplied with an axiomatic language definition satisfying our constraints, MetaVCG will automatically produce a VCG for the language.

After introducing some of the basic concepts to be used in this paper, we present the normal form for rules (Section 3), the less constrained rule form for user-defined rules (Section 4), and then the main soundness and completeness results (Section 5).

2. Background and Overview

2.1. The Basic Method

In 1969 Hoare [4] introduced the style of axiomatic semantics frequently used to define programming languages. Hoare's approach is to regard program text as specifying a relation between assertions. The notation $P\{S\}Q$ is used to mean that "if *precondition* P is true before execution of program fragment S, and if execution of S terminates, then *postcondition* Q is true upon its completion". The meaning of every simple statement (such as assignment) is defined by an axiom schema and every compound statement (such as composition) is defined by an inference rule schema. A logical system containing Hoare axiom and deduction schemas for all syntactic forms in a programming language constitutes a partial-correctness *axiomatic definition* or *axiomatization* of the language [8].

The role of a VCG in a program verification system is to reduce the correctness of a sentence $P\{S\}Q$ to the correctness of some number of lemmas, called *verification conditions*, in the underlying theory. The provability of these lemmas is intended to be sufficient to guarantee that an axiomatic proof using the Hoare system could be constructed.

Verification condition generation is typically performed using a recursively defined *predicate transformer* pre(S,Q), which maps a program fragment S and a postcondition Q into a precondition. The function pre computes an assertion *sufficient* to guarantee that Q will be derivable as a postcondition (i.e., that $\mathrm{pre}(S,Q)\{S\}Q$ is provable). The provability of the verification condition $P \supset \mathrm{pre}(S,Q)$ in the underlying logic is thus sufficient to show that $P\{S\}Q$ is provable within any Hoare system containing the rule of consequence. A predicate transformer that produces preconditions which are both *necessary* and *sufficient* to derive Q as the postcondition is said to compute the *weakest derivable precondition*, and is denoted wdp(S,Q).

Our notion of wdp should not be confused with Dijkstra's notion of weakest liberal precondition wlp [3]. Weakest in our context is with respect to provability from the axiomatic definition, while Dijkstra's notion of weakest is relative to an interpretive model.

We now show how to derive wdp from a Hoare axiomatization of a programming language. Suppose that the normal form characterization of rules is

$$\frac{P_1\{S_1\}\mathcal{Q}_1, \ldots, P_n\{S_n\}\mathcal{Q}_n, \Gamma}{\mathcal{P}\{S\}Q} \tag{1}$$

which permits the deduction of $\mathcal{P}\{S\}Q$ from the $n+1$ premises. For the moment, let each $\mathcal{Q}_i$, Γ, and $\mathcal{P}$ stand for arbitrary logical expressions involving predicate symbols $P_1,\ldots,P_n,Q$ and formulas in the underlying theory. Examples of rules of this form are the axiom for assignment (without side effects)

$$P[x \leftarrow e]\{x := e\}P \tag{2}$$

where $P[x \leftarrow e]$ indicates the proper substitution of the expression e for each occurence of the variable x in P, and the rule of inference for statement composition

$$\frac{P\{S_1\}R,\ R\{S_2\}Q}{P\{S_1;S_2\}Q} \tag{3}$$

Given any rule of form (1), wdp can be defined as follows:

$$\mathrm{wdp}(S,Q) = \mathcal{P}[P_1 \leftarrow \mathrm{wdp}(S_1,\mathcal{Q}_1), \ldots, P_n \leftarrow \mathrm{wdp}(S_n,\mathcal{Q}_n)] \land (\forall \bar{v})\, \Gamma\, [P_1 \leftarrow \mathrm{wdp}(S_1,\mathcal{Q}_1), \ldots, P_n \leftarrow \mathrm{wdp}(S_n,\mathcal{Q}_n)] \tag{4}$$

where $[P_1 \leftarrow t_1, \ldots, P_n \leftarrow t_n]$ denotes n proper substitutions carried out *sequentially* in a left-to-right order, and $\bar{v}$ is the set of all free logical variables in Γ. For example, the predicate transformer corresponding to assignment axiom (2) would be

$$\mathrm{wdp}(x:=e,P) = P[x \leftarrow e]$$

and the one corresponding to composition rule (3) would be

$$\mathrm{wdp}(S_1;S_2,Q) = P\,[P \leftarrow \mathrm{wdp}(S_1,R),\ R \leftarrow \mathrm{wdp}(S_2,Q)] = \mathrm{wdp}(S_1,\mathrm{wdp}(S_2,Q)) \qquad (5)$$

Notice that these are the predicate transformers usually associated with assignment and composition. In fact, the predicate transformers produced by wdp are the ones commonly used to mechanize Hoare logics.

2.2. The Two Main Problems

As just discussed, a VCG reduces the question of whether a sentence $P\{S\}Q$ is a theorem in Hoare logic to the question of whether $P \supset \mathrm{wdp}(S,Q)$ is a theorem in the underlying theory (e.g., first-order logic). Two important questions naturally arise:

1. **Soundness.** Does the VCG accurately reflect the semantics of the programming language as embodied by the associated Hoare logic? In other words, if $P \supset \mathrm{wdp}(S,Q)$ is provable in the underlying theory, is $P\{S\}Q$ provable in the Hoare logic?

2. **Completeness.** Is a VCG as "powerful" as the Hoare logic from which it is derived? In other words, if $P\{S\}Q$ is provable, is $P \supset \mathrm{wdp}(S,Q)$ also provable?

More formally, we must show that

$$\vdash_{\mathcal{H}} P\{S\}Q \qquad \text{iff} \qquad \vdash_{\mathcal{T}} P \supset \mathrm{wdp}_{\mathcal{H}}(S,Q)$$

where $\mathcal{H}$ is a Hoare axiom system, $\mathcal{T}$ is the underlying logical theory, $\mathrm{wdp}_{\mathcal{H}}(S,Q)$ is the predicate transformer derived from $\mathcal{H}$ as prescribed by (4), and $P \supset \mathrm{wdp}_{\mathcal{H}}(S,Q)$ is the formula in $\mathcal{T}$ produced by the VCG. This theorem does not hold for arbitrary $\mathcal{H}$, as explained later. Thus, the problem is to find a sufficient set of constraints on $\mathcal{H}$ that does not unreasonably restrict the expressiveness of the resulting logics. The general rule form constraints of Section 4 have this property.

2.3. A Unifying Model

We need a conceptual model that connects formal axiomatic proofs and VCGs based on wdp. Such a model is provided by viewing a VCG as an *automatic proof constructor* for Hoare logic. An axiomatic proof of a sentence $P\{S\}Q$ consists of a sequence of steps where the last step is $P\{S\}Q$, and each previous step is either an instance of an axiom schema, a theorem in the underlying logic, or follows from previous steps by applying an instance of a rule of inference. In our model, a VCG constructs such proofs, using wdp to find instantiations for free predicate symbols in axioms and rules of inference. For any sentence $P\{S\}Q$, the basic strategy is to instantiate precondition P with $\mathrm{wdp}(S,Q)$.

This model is illustrated below, where annotations (indicated by lines with roman numbering) relate the strategy used by the VCG in attempting to construct the formal proof of the sentence $\alpha\{z:=1;y:=z+1\}\beta$. Indentation indicates the nesting of recursive calls on wdp.

i. Select and instantiate composition rule for "$z:=1;y:=z+1$" with:
$S_1 \leftarrow z:=1$, $S_2 \leftarrow y:=z+1$, $Q \leftarrow \beta$, $R \leftarrow \mathrm{wdp}(y:=z+1,Q)=\beta[y \leftarrow z+1]$,
$P \leftarrow \mathrm{wdp}(z:=1,R)=\mathrm{wdp}(z:=1,\mathrm{wdp}(y:=z+1,\beta))=(\beta[y \leftarrow z+1])[z \leftarrow 1]$

ii. Apply assignment axiom for "$y:=z+1$" with:
$x \leftarrow y$, $e \leftarrow z+1$, $P \leftarrow \beta$
1. $\beta[y \leftarrow z+1]\{y:=z+1\}\beta$ assignment

iii. Apply assignment axiom for "$z:=1$" with:
$x \leftarrow z$, $e \leftarrow 1$, $P \leftarrow \beta[y \leftarrow z+1]$,
2. $(\beta[y \leftarrow z+1])[z \leftarrow 1]\ \{z:=1\}\ \beta[y \leftarrow z+1]$ assignment

3. $(\beta[y \leftarrow z+1])[z \leftarrow 1]\ \{z:=1;y:=z+1\}\ \beta$ composition (1,2)

4. $\alpha \supset (\beta[y \leftarrow z+1])[z \leftarrow 1]$ lemma

5. $\alpha\ \{z:=1;y:=z+1\}\ \beta$ consequence (4)

The overall proof strategy of the VCG is to select and instantiate the rule of inference that applies to the outermost syntactic structure in the program fragment (step i), satisfy its premises (steps ii and iii), and then conclude its conclusion (line 3). The VCG begins the proof by selecting the rule of composition (3) and performing the instantiations indicated above. This is a *valid instantiation* because it binds all free symbols in the rule. To see this, notice that when the substitution $[P \leftarrow \mathrm{wdp}(z:=1,R), R \leftarrow \mathrm{wdp}(y:=z+1,\beta)]$ is applied to the composition rule, we get $\mathrm{wdp}(y:=z+1,\beta)$ for R and $\mathrm{wdp}(z:=1,\mathrm{wdp}(y:=z+1,\beta))$ for P. Next, the VCG must prove both premises of the rule, namely $\mathrm{wdp}(y:=z+1,\beta)\{y:=z+1\}\beta$ and $\mathrm{wdp}(z:=1,\mathrm{wdp}(y:=z+1,\beta))\{z:=1\}\mathrm{wdp}(y:=z+1,\beta)$. Expanding the definition of wdp, we see that both are instances of assignment axiom (2), yielding lines 1 and 2 of the formal proof. Having satisfied the premises of the composition rule, the VCG concludes line 3 of the proof.

Lines 4 and 5 are instances of a two-line scheme that completes every proof done by the VCG. Line 4 corresponds to the formation of $P \supset \mathrm{wdp}(S,Q)$, which must be provable in the underlying theory for this to be a valid proof. Line 5 then follows immediately by the rule of consequence (ROC):

$$\frac{P \supset R,\quad R\{S\}T,\quad T \supset Q}{P\ \{S\}\ Q}$$

Although VCGs normally produce only line 4 as output, their output could just as easily be the entire axiomatic proof.

It should be pointed out that our model accurately describes the VCG only because of the restrictions we place on Hoare logics. The model clearly is inadequate for arbitrary Hoare logics. For example, it is easy to state a rule that requires the invention of an inductive assertion, which wdp is incapable of doing. The normal form constraints given in the next section were carefully chosen so that the VCG can always properly apply rules (consistency) and so that the instantiations computed by wdp will always be the weakest derivable instantiations (completeness).

This model provides the needed conceptual link between previous work on VCGs and Hoare logic. In the past, the VCG *itself* has been taken as the definition of the programming language. It is usually the case that the predicate transformer serves as the definition of each construct in the language, since there is often no axiomatic definition of the language. In some cases, however, there does exist an axiomatic definition of the programming language, but no formal correspondence between it and the predicate transformer is demonstrated. In both cases, we are left with the VCG as a *de facto* standard when automated proofs are attempted.

The only attempt at validating the view of a VCG as a proof constructor for an associated axiom system is the work of Igarashi, London, and Luckham [6]. In their paper, they give an axiomatic definition of a small language and an associated VCG. While they do not demonstrate a formal correspondence between their recursively specified VCG and their axiom system, they do prove that their axiom system is interderivable with another axiom system that more directly reflects the instantiation strategy employed by their VCG.

3. The Normal Form for Rules

This section defines a set of constraints on Hoare axioms and rules of inference under which the desired consistency and completeness property holds. Rules satisfying these constraints will be called *normal form rules.*

3.1. Notational Conventions and Preliminary Definitions

We will be defining properties of partially interpreted axiom and inference rule schemes, and must therefore carefully distinguish among three levels of discourse. In defining the normal form, we will use metavariables $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{R}$, ... (with or without subscripts) to denote partially interpreted, standard first-order formulas. These formulas can contain uninterpreted predicate symbols P, Q, R, ... (with or without subscripts) and formulas from the underlying theory. For example, $\mathcal{P}$ could denote P, $P \land x=5$, or $x=5$. We assume that uninterpreted predicate variables P, Q, R, ... may be instantiated by formulas in the underlying theory. For example, P could be instantiated as $x=5$, but not $Q \land x=5$.

We will make use of a binary relation $\Leftarrow$ on uninterpreted predicate symbols. For a Hoare sentence of the form

$$\mathcal{P}(P_1, ..., P_m)\ \{S\}\ \mathcal{Q}(Q_1, ..., Q_n)$$

where predicate symbols $P_1,...,P_m$ and $Q_1,...,Q_n$ are logically free in $\mathcal{P}$ and $\mathcal{Q}$, respectively, we have

$$P_i \Leftarrow Q_j \text{, for } i \in \{1,...,m\} \text{ and } j \in \{1, ..., n\}$$

Intuitively, a relation $P \Leftarrow Q$ should be thought of as indicating that the binding of predicate symbol P is dependent upon the binding of predicate symbol Q. The relation $\Leftarrow$ is defined with respect to a set of Hoare sentences in the obvious way. The notation $\overset{+}{\Leftarrow}$ denotes the *transitive closure* of $\Leftarrow$. Whenever we have $H \overset{+}{\Leftarrow} T$, H will be called the *head* of the dependency chain and T the *tail.*

Similarly, for a rule of the form given in (1), we employ the relation $\ll$ to define the dependence of a proof concerning S on proofs concerning $S_1, ..., S_n$. In particular, we have $S \ll S_i$, for $i \in \{1,...,n\}$. For a Hoare axiom system, we define the transitive closure $\ll +$ in the obvious manner.

We use the function FreePreds to denote the set of *logically free* predicate symbols in a formula, a Hoare sentence, or a Hoare rule. FreePreds applied to a formula denotes its logically free symbols. FreePreds applied to a Hoare sentence $\mathcal{P}\{S\}\mathcal{Q}$ is simply the union of FreePreds($\mathcal{P}$) and FreePreds($\mathcal{Q}$), and FreePreds applied to an inference rule is the union of the predicate symbols obtained by applying FreePreds to each premise and the conclusion of the rule.

We will use the function FragVars to denote the set of "fragment variables" in the language fragment S of a Hoare sentence P{S}Q. For example, FragVars applied to "**if** B **then** S_1 **else** S_2 **fi**" has the value $\{B, S_1, S_2\}$. If applied to an entire Hoare rule, FragVars yields a set containing the fragment variables from every Hoare sentence in the rule.

Lastly, we use these two functions in defining the notion of a bound occurrence of an uninterpreted predicate symbol in a rule. For a rule R, a predicate symbol in FreePreds(R) is *bound in R* if and only if it is in FragVars(R). Otherwise, the occurrence is said to be *free in R*. Intuitively, we are carefully distinguishing those logically free variables that are bound in the program fragment when a rule is applied (i.e., those bound in the rule) from those that must be bound by wdp.

3.2. The Constraints

We will state the complete definition of the normal form and then explain it in detail.

The Normal Form

A *normal form rule* is any instance N of

$$\frac{P_1\{S_1\}\mathcal{Q}_1\ ,\ldots,\ P_n\{S_n\}\mathcal{Q}_n\ ,\ \Gamma}{\mathcal{P}\{S\}Q}$$

that satisfies the following constraints:

1. $P_1, \ldots, P_n$ and Q are predicate symbols free in N.

2. Γ is a sentence in the underlying theory whose logically free predicate symbols can include only those in FreePreds(N) or FragVars(S).

3. The fragment variables of each S_i in the premises must be bound in S. That is, it must be the case that $\bigcup_{1 \leq i \leq n} \mathrm{FragVars}(S_i) \subseteq \mathrm{FragVars}(S)$.

4. *Dependency ordering.* The Hoare-sentence premises of N must satisfy two dependency constraints.

   a. $P_i \overset{\pm}{=} P_j \supset i < j$

   b. $T \overset{\pm}{=} U \wedge \neg(\exists R) U \overset{\pm}{=} R \supset U \equiv Q \vee U$ bound in N

5. *Monotonicity.* Let $\mathcal{P}[P \leftarrow \mathrm{false}, P \in s]$ denote $\mathcal{P}$ with the proper substitution of false for each predicate P in the set s. Then, the following constraint on $\mathcal{P}$ must be satisfied:

$$\mathcal{P}[P_1,\ldots,P_n,Q \leftarrow \mathrm{true}] \ \vee\ \forall s \subseteq \{P_1,\ldots,P_n,Q\}\ \neg\mathcal{P}[P \leftarrow \mathrm{false}, P \in s])$$

This constraint must hold for Γ (with $\mathcal{P}$ replaced by Γ) and for each $\mathcal{Q}_i$ (with $\mathcal{P}$ replaced by $\mathcal{Q}_i$).

For axioms, this definition collapses to sentences of the form $\mathcal{P}(Q)\{S\}Q$, where postcondition Q is the only predicate symbol that can be free in the axiom and the following constraint must be satisfied: $\mathcal{P}[Q \leftarrow \mathrm{true}] \vee \neg\mathcal{P}[Q \leftarrow \mathrm{false}]$.

Two constraints are placed on a collection of normal form rules: (i) Any terminal string σ in the programming language can be an instance of at most one language fragment S defined by by a normal form axiom or inference rule. (ii) The relation $\ll+$ must be irreflexive. (We show later that this will guarantee termination of wdp.) Also, accompanying every normal form system are the ROC and the axiom $\mathrm{false}\{S\}Q$.

Constraints 4 and 5 require further explanation. Constraint 4 ensures that wdp will be able to compute instantiations for all free, uninterpreted predicate symbols in rules using left-to-right substitution of $\mathrm{wdp}(S_i,\mathcal{Q}_i)$ for each $\mathcal{P}_i$. This is done by first imposing restrictions on where free predicate symbols can occur in rules, and then placing constraints on some of these symbols based on dependency considerations. Constraint 4a requires an ordering of free predicate symbols that is made apparent by the following schema:

$$\frac{P_1\{S_1\}\mathcal{Q}_1(P_2,...,P_n),\ ...,\ P_i\{S_i\}\mathcal{Q}_i(P_{i+1},...,P_n),\ ...,\ P_n\{S_n\}\mathcal{Q}_n}{\mathcal{P}(P_1,...,Q)\,\{S\}\,Q}$$

This says that every precondition of a Hoare sentence premise of an inference rule can depend only upon preconditions occurring in subsequent premises. This has the effect of eliminating dependency cycles, such as a premise of the form P{...}P (as in the case of the "unasserted" while statement) or a pair of premises of the form P{...}R and R{...}P. In neither case would wdp be able to find an instantiation for the repeated symbol P. Also note that 4a requires not only that a proper ordering of premises exists, but that premises actually be placed in the prescribed order. For example, if the premises for of composition rule (3) were reversed, it would not satisfy 4a.

Given this ordering, constraint 4b ensures that the tail of every dependency chain is either expressible as a function of postcondition Q or is bound in a program fragment. In the hypothesis of 4b, U is the tail of a dependency chain $T\overset{+}{\Leftarrow}U$ which does not also occur as the head of another chain (i.e., there is no other R such that $U\overset{+}{\Leftarrow}R$). The conclusion of 4b says that every such U must be either the postcondition Q or a fragment variable in N, both of which are bound without the use of wdp when a rule is applied. Composition rule (3), for example, satisfies this constraint, since postcondition Q is the only tail not also occurring as the head of a dependency chain. In contrast, a rule containing premises P{...}T , S{...}R , and R{...}Q would not be allowed, unless T were bound in a program fragment. Otherwise, wdp would not compute an instantiation for T.

Constraint 5 is necessary for completeness. Recall that the completeness of a VCG hinges upon its ability to compute the weakest derivable precondition wdp(S,Q) for a given S and Q. As the simplest example of a rule for which wdp cannot compute the weakest derivable precondition, consider the axiom ¬Q{S}Q . From this axiom, it is possible to prove true{S}true using the ROC. The predicate transformer associated with this axiom by (4) is wdp(S,Q) = ¬Q , meaning that wdp(S,true) = false . But true (not false) is the weakest derivable precondition. This same sort of difficulty can result from interactions among several different rules.

Therefore, Constraint 5 imposes a monotonicity constraint on rules, which eliminates rules in which certain "changes of sign" exist between the preconditions of the premises and the precondition in the conclusion. The first disjunct of 5 says that an inference rule that does not have a sign change is acceptable. That is, if the truth of $\mathcal{P}$ follows from the truth of $P_1, ..., P_n$ and Q, the rule is acceptable. The second disjunct states that a sign change in an inference rule is acceptable if the falsity of $\mathcal{P}$ is *independent* of the free variables in the rule. More precisely, it says that a rule is acceptable if there are no truth values assignments to $P_1, ..., P_n$ and Q that will make $\mathcal{P}$ true. Whenever this is the case, we know that any sign change is a function of predicate symbols bound in the language fragment; it turns out that this does not result in incompleteness. The axiom ¬Q{S}Q above does not satisfy this constraint.

A normal form definition of a simple language is given in Figure 1; the general form of this definition is given in the next section. Although the while rule N4 and the conditional rule N5 may appear unusual, their general rule form is the common one. Also note that the procedure declaration and call rules (N7 and N8, respectively) use assertion-language functions to handle the association between procedure declaration and call. The predicate boundP(p,Q) is used in N8 to test whether there is an expression of the form bind(p,⟨*assertion,assertion,variable*⟩) in Q before total functions getpre, getpost, and getvars are applied to retrieve binding information at the point of call. A more elegant approach to handling this contextual information is suggested in the conclusion. For pedagogical reasons, we assume in our simple language that expressions have no side effects, procedures are nonrecursive, procedure as parameters and aliasing in procedure calls are prohibited, and global variables are disallowed.

Axioms

N1. simple assignment: $P[x \leftarrow e]\ \{x := e\}\ P$

N2. array assignment: $P[a \leftarrow \mathrm{arrayUpdate}(a, e_1, e_2)]\ \{a[e_1] := e_2\}\ P$

N3. empty statement: $P\{\ \}P$

Rules of inference

N4. iteration:

$$\frac{P_1\{S\}P, \quad P \wedge \neg B \supset Q, \quad P \wedge B \supset P_1}{P\ \{\textbf{while}\ B\ \textbf{assert}\ P\ \textbf{do}\ S\ \textbf{od}\}\ Q}$$

N5. conditional:

$$\frac{P_1\{S_1\}Q, \quad P_2\{S_2\}Q}{B \supset P_1 \wedge \neg B \supset P_2\ \{\textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\}\ Q}$$

N6. compound statements:

$$\frac{P\{S_1\}R, \quad R\{S_2\}Q}{P\{S_1;\ S_2\}\ Q}$$

N7. procedure declaration:

$$\frac{U\{S_1\}Q, \quad R\{S_2\}T \wedge \mathrm{bind}(p, \langle P, Q, x \rangle), \quad P \supset U}{R\{\textbf{begin proc}\ p(\textbf{var}\ x) = \textbf{pre}\ P;\ \textbf{post}\ Q;\ S_1\ \textbf{corp};\ S_2\ \textbf{end}\}T}$$

N8. procedure call:

$$\frac{\mathrm{boundP}(p, Q)}{(\forall \bar{v})(\mathrm{getpre}(p,Q)[\mathrm{getvars}(p,Q) \leftarrow a] \wedge \mathrm{getpost}(p,Q)[\mathrm{getvars}(p,Q) \leftarrow x']) \supset Q[a \leftarrow x']\ \{p(a)\}\ Q}$$

Figure 1: Example normal form language definition.

4. An Equivalent Rule Form With Fewer Constraints

So far, we have explained the normal form for rules and how to transform them into a VCG. This section presents the remaining part of our method, which is motivated by the practical concern of wanting to impose as few constraints as possible on rules written by users of MetaVCG. The general rule form defined below allows considerable flexibility in stating premises to inference rules -- premises need not be ordered and may have more general preconditions. This rule form has the important property that any rule satisfying its constraints can be mechanically transformed into an equivalent normal form rule. The normal form rules of Figure 1 that are more conveniently expressed in this general rule form are contained in Figure 2, and the transformation between the two rule forms is defined in the appendix.

The General Rule Form

A *general form rule* is any instance G of

$$\frac{\mathcal{S}_1, \ldots, \mathcal{S}_n, \Gamma}{\mathcal{P}\{S\}\ Q}$$

that satisfies normal form constraints 1-3 and 4b, where:

1. Each premise $\mathcal{S}$ is a Hoare sentence of one of the following forms.

a. $\mathcal{R}\ \{S\}\ \mathcal{Q}$ b. $\mathcal{F}\ \{S\}\ \mathcal{Q}$ c. $\mathcal{R} \wedge \mathcal{F}\ \{S\}\ \mathcal{Q}$

where, in all three cases, $\mathcal{R}$ is a metavariable evaluating to a single predicate symbol free in G, $\mathcal{F}$ is a metavariable evaluating to a formula not containing any predicate symbols free in G, and $\mathcal{Q}$ is a metavariable.

2. The relation $\stackrel{+}{=}$ must be irreflexive with respect to $\mathcal{S}_1, \ldots, \mathcal{S}_n$.

3. Let $\bar{r}$ be the set of predicate symbols free in the preconditions of $\mathcal{S}_1, \ldots, \mathcal{S}_n$. Then, the following constraint on $\mathcal{P}$ must hold:

$$\mathcal{P}[P \leftarrow \mathrm{true},\ P \in \bar{r} \cup \{Q\}] \quad \vee \quad \forall s \subseteq \bar{r} \cup \{Q\}\ \neg\mathcal{P}[P \leftarrow \mathrm{false},\ P \in s]$$

This constraint must hold for Γ (with $\mathcal{P}$ replaced by Γ) and for each $\mathcal{Q}_i$ (with $\mathcal{P}$ replaced by $\mathcal{Q}_i$).

G1. Iteration:

$$\frac{P\wedge B\{S\}P,\ P\wedge\neg B\supset Q}{P\ \{\textbf{while}\ B\ \textbf{assert}\ P\ \textbf{do}\ S\ \textbf{od}\}\ Q}$$

G2. Conditional:

$$\frac{P\wedge B\{S_1\}Q,\ P\wedge\neg B\{S_2\}Q}{P\ \{\textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\}\ Q}$$

G3. Procedure declaration:

$$\frac{P\{S_1\}Q,\quad R\{S_2\}T\wedge bind(p,\langle P,Q,x\rangle)}{R\{\textbf{begin proc}\ p(\textbf{var}\ x)=\textbf{pre}\ P;\ \textbf{post}\ Q;\ S_1\ \textbf{corp};\ S_2\ \textbf{end}\}T}$$

Figure 2: Acceptable renditions of awkward normal form rules.

The interesting constraints are the first two. Constraint 1 gives the user considerable flexibility in expressing the Hoare sentence premises of an inference rule by lifting three normal form restrictions. Constraint 1a allows duplicate free predicate symbols as preconditions, and 1c allows a combination of (possibly duplicate) free and bound predicate symbols. Rules G1 and G2 illustrate the utility of this weakening of the normal form constraints. Constraint 1b allows preconditions whose logically free variables are bound in the rule, as illustrated by G3.

Constraint 2 is the only dependency constraint. It says that the Hoare sentence premises of an inference rule can be unordered, provided there are no dependency cycles. This is in contrast to normal form constraint 4a, which requires a very particular ordering of premises.

The collection of general form axioms and rules of inference must satisfy the two overall constraints given for the normal form system.

5. Formal Basis for the Method

To demonstrate that a VCG constructed by our method is sound and deduction-complete with respect to a general form axiomatic definition $\mathcal{G}$, we prove the following theorem.

> **Theorem:** Let $\mathcal{G}'$ be any general form axiom system $\mathcal{G}$ augmented by the rule of consequence and the axiom false{S}Q , and let τ denote the transformation from $\mathcal{G}$ to the normal form, and suppose that $\mathcal{T}$ is a complete (perhaps noneffective) proof system for the underlying theory. Then
>
> $$\vdash_{\mathcal{G}'} P\{S\}Q \quad \text{iff} \quad \vdash_{\mathcal{T}} P\supset wdp_{\tau(\mathcal{G})}(S,Q)\ .$$

The proof is done in two steps, showing first that

$$\vdash_{\mathcal{G}'} P\{S\}Q \quad \text{iff} \quad \vdash_{\tau(\mathcal{G})'} P\{S\}Q$$

and then that

$$\vdash_{\mathcal{N}'} P\{S\}Q \quad \text{iff} \quad \vdash_{\mathcal{T}} P\supset wdp_{\mathcal{N}}(S,Q)$$

where $\mathcal{N}'$ is any normal form axiom system $\mathcal{N}$ augmented by the ROC and the axiom false{S}Q . The former lemma demonstrates that a general form system $\mathcal{G}$ is sound and deduction-complete with respect to the normal form representation of $\mathcal{G}$ under τ. Its proof is tedious but routine and will not be given here. The second lemma, which we prove here, establishes that VCGs constructed by our method are sound and deduction-complete with respect to any normal form system $\mathcal{N}'$.

When wdp(S,Q) appears in a formula, there is an implicit assertion that it terminates and denotes a formula in the underlying theory. As part of the completeness proof, we will prove that, whenever a sentence P{S}Q is provable in $\mathcal{N}'$, wdp(S,Q) always in fact terminates and produces a formula in $\mathcal{T}$.

5.1. Main Soundness Result

In this section we prove the consistency of a VCG with respect to its associated normal form axiom system augmented by the ROC. The soundness lemma to be proved is:

$$\text{If } \vdash_{\mathcal{T}} P \supset wdp_{\mathcal{N}'}(S,Q) \text{ then } \vdash_{\mathcal{N}'} P\{S\}Q \ .$$

Henceforth, we will usually omit explicit reference to theories and will use wdp(S,Q) is an abbreviation for $wdp_{\mathcal{N}'}(S,Q)$.

The proof is by induction on the depth of recursive application of wdp. In terms of our proof constructor model, we must show that wdp properly applies the axioms and rules of inference defining each construct of the programming language. Each recursive application of wdp must correspond to a valid proof step. We show that, for each S defined by an axiom, wdp(S,Q){S}Q is provable, and that for each S defined by an inference rule, wdp(S,Q){S}Q is provable whenever $wdp(S_i,\mathcal{Q}_i)\{S_i\}\mathcal{Q}_i$ is provable for each premise of the rule. This demonstrates that wdp(S,Q){S}Q holds for any construct S; the hypothesis and the ROC can then be used to obtain the desired conclusion.

As the base case for the induction, we consider the situation in which S is defined by an axiom of the form $\mathcal{P}(Q)\{S\}Q$. By the definition of wdp, we get wdp(S,Q){S}Q , from which the desired conclusion follows.

We now show that wdp properly applies inference rules defining the composite constructs of the language. This means that, for any normal form inference rule N, (i) wdp must find a valid instantiation of N and (ii) if $wdp(S_i,\mathcal{Q}_i)$ finds a valid instantiation for each of the n Hoare-sentence premises, then wdp(S,Q){S}Q follows. To establish (i) we must show that the left-to-right substitution

$$[P_1 \leftarrow wdp(S_1,\mathcal{Q}_1), \ldots, P_n \leftarrow wdp(S_n,\mathcal{Q}_n)] \tag{6}$$

binds all free predicate symbols in N. Recall that the premise of a normal form inference rule consists of n Hoare-sentence premises of the form $P_i\{S_i\}\mathcal{Q}_i$ and a sentence Γ in the underlying theory. Normal form constraints 4a and 4b require that each $\mathcal{Q}_i$ contain as logically free predicate symbols only $P_{i+1},\ldots,P_n$, Q or predicate symbols bound in S. Further, these are the only symbols that can be free in $wdp(S_i,\mathcal{Q}_i)$. The successive left-to-right substitutions given in (6) will then eliminate each P_i in the Hoare-sentence premises. This leaves as logically free predicate symbols only Q and those bound in S, all of which are bound whenever a rule is applied. It follows from normal form constraint 2 that (6) also binds all free symbols in Γ.

We next establish (ii). We take as inductive hypotheses

$$wdp(S_1,\mathcal{Q}_1)\{S_1\}\mathcal{Q}_1, \ldots, wdp(S_n,\mathcal{Q}_n)\{S_n\}\mathcal{Q}_n \tag{7}$$

i.e., that wdp generates valid preconditions for each Hoare-sentence premise. Now let $\mathcal{Q}_i'$, $\mathcal{P}'$, and Γ' stand for $\mathcal{Q}_i$, $\mathcal{P}$, and Γ under (6). More specifically, $\mathcal{Q}_i'$ is

$$\mathcal{Q}_i\,[P_{i+1} \leftarrow wdp(S_{i+1},\mathcal{Q}_{i+1}), \ldots, P_n \leftarrow wdp(S_n,\mathcal{Q}_n)] \ , \tag{8}$$

$\mathcal{P}'$ is

$$\mathcal{P}[P_1 \leftarrow wdp(S_1,\mathcal{Q}_1), \ldots, P_n \leftarrow wdp(S_n,\mathcal{Q}_n)] \ ,$$

and Γ' is defined analogously to $\mathcal{P}'$. From our previous analysis of dependency constraints, we know that the only free predicate symbol in $\mathcal{Q}_i'$ is Q. $\mathcal{Q}_i'$ is thus a valid instantiation for $\mathcal{Q}_i$. Using this instantiation and our inductive hypothesis (7), we can conclude

$$\mathrm{wdp}(S_1,Q_1')\{S_1\}Q_1', \ldots, \mathrm{wdp}(S_n,Q_n')\{S_n\}Q_n' \quad . \tag{9}$$

We now show that our VCG is sound independent of whether $(\forall\bar{v})\Gamma'$ is provable in $\mathcal{T}$. First suppose that $\vdash(\forall\bar{v})\Gamma'$. This coupled with (9) satisfies all the premises of N, allowing us to conclude $\mathcal{P}'\{S\}Q$, from which we obtain $\mathcal{P}'\wedge(\forall\bar{v})\Gamma'\{S\}Q$, which according to the definition of wdp is the same as $\mathrm{wdp}(S,Q)\{S\}Q$. Now assume that $\neg(\forall\bar{v})\Gamma'$. Then, from the definition of wdp, we see that $\mathrm{wdp}(S,Q)\supset\mathrm{false}$. In this case we can use the axiom $\mathrm{false}\{S\}Q$ and the ROC to conclude $\mathrm{wdp}(S,Q)\{S\}Q$.

The above induction argument shows that $\mathrm{wdp}(S,Q)\{S\}Q$ is provable for all S. The final step is to observe that our assumption that $P\supset\mathrm{wdp}(S,Q)$ and the ROC can now be used to conclude $\vdash_{\mathcal{N}'} P\{S\}Q$. ∎

5.2. Main Completeness Result

In this section we prove the completeness of a predicate transformer system produced by our method with respect to the sentences derivable from the axiom system $\mathcal{N}'$. In particular, we prove that

$$\vdash_{\mathcal{N}'} P\{S\}Q \quad \text{implies} \quad \vdash_{\mathcal{T}} P\supset\mathrm{wdp}(S,Q) \tag{10}$$

The proof is by induction on the number of recursive calls on wdp. We must show that, for any provable sentence $P\{S\}Q$, wdp can construct a proof using the weakest derivable instantiations. As the base case for our induction, we show that wdp computes the weakest derivable preconditions when S is defined by an axiom. The induction step considers the situation in which S is defined by a rule of inference. We prove inductively that if wdp generates the weakest precondition for each Hoare-sentence in the premises of the rule, then it will generate the weakest derivable precondition for the S defined by the rule. This is sufficient to show establish our theorem, since each premise used in the application of an inference rule is deduced from application of another inference rule or follows from an axiom.

Before presenting the main proof, we first establish that the function $\mathrm{wdp}(S,Q)$ always terminates. If S is defined by an axiom, wdp terminates because it involves no recursion. For S defined by an inference rule N, the termination of $\mathrm{wdp}(S,Q)$ depends upon the termination of each $\mathrm{wdp}(S_i,Q_i)$ in the n Hoare-sentence premises of N. Our overall system constraint that $\ll+$ is irreflexive guarantees that the proof of a sentence concerning S cannot depend upon satisfying a premise concerning S. The sequence of inferences attempting to satisfy the premises of N must therefore be finite, and thus the computation of $\mathrm{wdp}(S,Q)$ must also be finite.

Case 1 (*S defined by axiom*). Our theorem clearly holds if S is proved using $\mathrm{false}\{S\}Q$ (since P must be false). Now suppose S is proved using normal form axiom $\mathcal{P}(Q)\{S\}Q$, whose corresponding predicate transformer is $\mathrm{wdp}(S,Q)=\mathcal{P}(Q)$. Since this axiom uniquely defines S (excluding $\mathrm{false}\{S\}Q$ from consideration), it must be applied in any proof of $P\{S\}Q$. In the most general setting, a proof of $P\{S\}Q$ would involve showing that $P\supset\mathcal{P}(R)$ and $R\supset Q$, and then using the instance $\mathcal{P}(R)\{S\}R$ of our axiom and the ROC to conclude $P\{S\}Q$. Thus, our theorem (10) holds for axioms if we can show that $P\supset\mathrm{wdp}(S,R)\supset\mathrm{wdp}(S,Q)$.

We first observe that $P\supset\mathrm{wdp}(S,R)$ follows from the definition of wdp for S and the fact that $P\supset\mathcal{P}(R)$. We now show that $\mathrm{wdp}(S,R)\supset\mathrm{wdp}(S,Q)$ -- or equivalently $\mathcal{P}(R)\supset\mathcal{P}(Q)$ -- follows from the fact that $R\supset Q$ and the monotonicity normal form constraint. There are two ways in which $R\supset Q$ can hold. If R is true, $\mathcal{P}(R)\supset\mathcal{P}(Q)$ clearly holds since Q must also be true. Now assume that R is false and $\mathcal{P}(R)$ is true. Since our monotonicity constraint requires that $\mathcal{P}(\mathrm{true}) \vee \neg\mathcal{P}(\mathrm{false})$, $\mathcal{P}(\neg R)$ must also be true. Hence, the truth of $\mathcal{P}$ is thus independent of the truth value of the free predicate symbol, and $\mathcal{P}(Q)$ must also be true. Thus wdp(S,Q) is the weakest derivable precondition if S is defined by an axiom. ∎

Case 2 (*S defined by a rule of inference*). Suppose that S is defined by normal form inference rule N. Our inductive hypothesis asserts that, from postcondition Q, wdp generates the weakest derivable instantiation for each precondition in the premises of N. That is, we assume that

$$P_1 \supset \mathrm{wdp}(S_1, \mathcal{Q}'_1(Q)) \ ,\ldots,\ P_n \supset \mathrm{wdp}(S_n, \mathcal{Q}'_n(Q)) \tag{11}$$

where $\mathcal{Q}'_i$ is defined as before (8) and will contain only postcondition Q as a free variable.

We begin by observing that any proof of $P\{S\}Q$ must necessarily follow a certain pattern, and then prove inductively that corresponding to any such proof is a proof of $P \supset \mathrm{wdp}(S,Q)$ in $\mathcal{T}$. The general form of any proof of $P\{S\}Q$ using inference rule N must proceed as follows: Since S can be defined only by N, we know that instantiating predicate symbols $P_1, \ldots, P_n$ and postcondition Q will yield a sentence of the form $R\{S\}U$ such that $P \supset R$ and $U \supset Q$. The ROC would then be used to conclude $P\{S\}Q$. This argument can be characterized more formally as applying some substitution $[P_1 \leftarrow R_1, \ldots, P_n \leftarrow R_n, Q \leftarrow U]$ to N to obtain

$$\frac{R_1\{S_1\}T_1, \ldots, R_n\{S_n\}T_n\,, \gamma}{R\{S\}U}$$

such that the premises $R_1\{S_1\}T_1, \ldots, R_n\{S_n\}T_n$ and γ are satisfied and where $P \supset R$ and $U \supset Q$. Using the ROC, we conclude $P\{S\}Q$.

This observation guides the subsequent argument, which shows inductively that $R \supset \mathrm{wdp}(S,U)$ given that $P \supset R$. A similar argument can be used to show that $\mathrm{wdp}(S,U) \supset \mathrm{wdp}(S,Q)$ given that $U \supset Q$. Combining these arguments yields $P \supset \mathrm{wdp}(S,Q)$, which means that our method is complete for S defined by an inference rule.

We begin by instantiating our inductive hypothesis (11) with $[P_1 \leftarrow R_1, \ldots, P_n \leftarrow R_n, Q \leftarrow U]$, yielding

$$R_1 \supset \mathrm{wdp}(S_1, \mathcal{Q}'_1(U)) \ ,\ldots,\ R_n \supset \mathrm{wdp}(S_n, \mathcal{Q}'_n(U)) \tag{12}$$

We proceed to show that wdp computes the weakest derivable precondition for S and Q, i.e.,

$$\vdash R \ \supset\ \mathrm{wdp}(S,U) \tag{13}$$

Noting that R is obtained by the proper substitution of $R_1, \ldots, R_n$ and U into precondition $\mathcal{P}$ of N, and expanding the definition of wdp, we will prove (13) by showing in two independent steps that

$$\mathcal{P}[P_1 \leftarrow R_1, \ldots, P_n \leftarrow R_n, Q \leftarrow U] \ \supset\ \mathcal{P}[P_1 \leftarrow \mathrm{wdp}(S_1, \mathcal{Q}_1), \ldots, P_n \leftarrow \mathrm{wdp}(S_n, \mathcal{Q}_n), Q \leftarrow U] \tag{14}$$

and that

$$\gamma \ \supset\ (\forall \bar{v})\Gamma\,[P_1 \leftarrow \mathrm{wdp}(S_1, \mathcal{Q}_1), \ldots, P_n \leftarrow \mathrm{wdp}(S_n, \mathcal{Q}_n), Q \leftarrow U] \quad . \tag{15}$$

Choose any assignment of truth values to $R_1, \ldots, R_n$ such that the antecedent of (14) holds. For any true R_i, $\mathrm{wdp}(S_i, \mathcal{Q}'_i(U))$ must be true by the inductive hypothesis. Hence, $\mathcal{P}[P_1 \leftarrow R_1, \ldots, P_{i-1} \leftarrow R_{i-1}, P_i \leftarrow \mathrm{wdp}(S_i, \mathcal{Q}'_i(U)), P_{i+1} \leftarrow R_{i+1}, \ldots, P_n \leftarrow R_n, Q \leftarrow U]$ must also be true.

Now consider any false R_i such that the antecedent of (14) holds. Recall that our monotonicity constraint requires

$$\mathcal{P}[P_1, \ldots, P_n, Q \leftarrow \mathrm{true}] \ \lor\ \forall s \subseteq \{P_1, \ldots, P_n, Q\} \ \neg\mathcal{P}[P \leftarrow \mathrm{false}, P \in s] \quad .$$

The first disjunct must hold (since, by hypothesis, there exists a false interpretation of an R_i rendering $\mathcal{P}$ true). But this implies that $\mathcal{P}$ is true irrespective of the truth value of R_i. Therefore, $\mathcal{P}[P_i \leftarrow \mathrm{wdp}(S_i, \mathcal{Q}'_i(U))]$ will be true irrespective of the truth value of $wdp(S_i,\mathcal{Q}'_i(U))$. This completes the proof of (14). The proof of (15) is exactly analogous, since Γ may contain only the free predicate symbols assumed in the proof of (14) and must satisfy an analogous monotonicity constraint.

This completes the proof of $R \supset wdp(S,U)$ at (13). Recall that, in order to complete the entire proof, we must consider derivations of $P\{S\}Q$ that use the second half of the ROC. Specifically, we must show that, given the fact that $U \supset Q$, $wdp(S,U) \supset wdp(S,Q)$. The requires an inductive argument following the reasoning used in the preceding one for (13). Expanding the definitions of $wdp(S,U)$ and $wdp(S,Q)$, we can show that each $wdp(S_i,\mathcal{Q}'_i(U)) \supset wdp(S_i,\mathcal{Q}'_i(Q))$, and thus that $wdp(S,U) \supset wdp(S,Q)$. Our hypothesis that $P \supset R$ and the fact that $R \supset wdp(S,U)$ can be used to conclude $P \supset R \supset wdp(S,U) \supset wdp(S,Q)$, thereby completing our proof of Case 2. Combining this with Case 1 demonstrates the completeness of of wdp with respect to the augmented normal form axiom system $\mathcal{N}'$. ▮

6. Conclusion and Future Work

The practical significance of this work is that it is now possible to correctly mechanize a useful class of Hoare logics by automated means. A VCG constructed by our method can serve not only as a central component of a program verification system, but it can also serve as a vehicle for "debugging" axiomatic definitions and exploring the semantic effect of various language design decisions.

It should be pointed out that a VCG produced by our method is correct *only if* the axiomatic definition of the programming language is correct. Thus, the remaining step in validating a verification system as a basis for reasoning about program behavior is to prove that the axiomatic definition is consistent and relatively complete with respect to an interpretive (operational) language model, as done by [2, 1].

Our theoretical results demonstrate that the traditional VCG paradigm is correct when certain constraints are placed on the rule forms in the associated Hoare logic. In addition, we found that MetaVCG's soundness depends upon the presence of $false\{S\}Q$ as an axiom, which brings out a interesting anomaly in the commonly used VCG paradigm. Recall that wdp is constructed from an inference rule by conjoining the premise Γ in the underlying theory to the precondition in the conclusion. In essence, Γ is "carried back" through the proof by wdp rather than occurring as a proof step to justify the application of the inference rule. As a simple example, instead of applying a rule with premise Γ and conclusion $\mathcal{P}\{S\}Q$, the VCG in effect applies the axiom $\mathcal{P} \wedge (\forall \bar{v})\Gamma\{S\}Q$. However, moving Γ from the premise to the precondition in the conclusion is not in general valid. While nothing can be proven from the original inference rule if Γ is unsatisfiable, $wdp(S,Q)\{S\}Q$ can be proven from the rule which the VCG actually applies. This is sound only if $false\{S\}Q$ is independently provable. Clearly, attempting to prove $false\{S\}Q$ when S is defined by an inference rule with an unsatisfiable premise is pathological. Nevertheless, it does illustrate that the paradigm only works if either $false\{S\}Q$ is an axiom or, for every inference rule, all premises in the underlying theory are satisfiable.

The expressiveness of our present theory is in principle sufficient for defining the semantics of real programming languages, provided the assertion language is rich enough. However, as a practical matter, it lacks the expressive power necessary to deal adequately with "context-dependent" semantics, such as full static scope, aliasing, side effects, exceptions, and procedures as parameters. Our method does well with context-independent properties of language semantics, but transfers much of the burden for defining context-dependent semantics to functions embedded in the assertion language.

We are exploring several extensions to our method, all subject to the constraint that they preserve its soundness and completeness. We are investigating the use of an enriched Hoare sentence C/P{S}Q, as described in [1, 10]. The additional C component expresses static information concerning program structure. This would allow us to reason about context directly within the Hoare logic, rather than having to embed contextual information in the assertion language (as was done in procedure declaration and call rules G3 and N8). Employing the context component, the revised procedure rules might be of the form:

$$\frac{C \cup \{\langle p, \mathrm{proc}(x)\,\mathrm{pre}\ P, \mathrm{post}\ Q\rangle\} \,/\, P\{S_1\}Q, \quad R\{S_2\}T}{C \,/\, R\{\mathbf{begin\ proc(var}\ x) = \mathbf{pre}\ P;\ \mathbf{post}Q;\ S_1\ \mathbf{corp};\ S_2\ \mathbf{end}\}T}$$

$$C \cup \{p, \mathrm{proc}(x)\,\mathrm{pre}\ R, \mathrm{post}\ T\rangle\} \,/\, (\forall \bar{v})(R[x \leftarrow a] \wedge T[x \leftarrow x']) \supset Q[a \leftarrow x']\ \{p(a)\}\ Q$$

The use of the context component in the first rule allows us to define that the context for elaboration of the block body S_2 and procedure body S_1 is the surrounding context C augmented by the local block declaration for procedure p. The procedure call axiom then defines that the meaning of a procedure call is determined from the context at the point of call. Recursive procedures are handled by these rules.

Our current constraints allow only rules that have an inherent "backward" orientation and to which a backward predicate transformer semantics can be assigned. Our theory (and implementation) could be adapted to handle forward-oriented rules with a forward predicate transformer semantics as well. Based on analysis of predicate dependencies, MetaVCG could choose the appropriate substitution direction, provided that all of the rules have the same orientation. For example, the forward-oriented Algol 68 axiomatization [9] could be handled. Further extensions to handle axiom systems with no consistent orientation are also being studied.

Finally, we are exploring methods of introducing early interpretation of functions in the underlying theory to allow, for example, for interleaved generation and simplification of verification conditions. At present, all functions in the underlying theory (including proper substitution) remain uninterpreted throughout verification condition generation.

Acknowledgments. We would like to thank Larry Flon who, together with the first author, developed the initial concept of MetaVCG. Valuable comments on this paper were provided by Wolf Polak, Bernie Elspas, Pepe Meseguer, Steve Crocker, and Joe Goguen. This work was supported in part by NASA-Langley Research Center under Contract NAS1-15528 and Rome Air Development Center under Contract F30602-78-C-0031.

References

1. E.M. Clarke, Jr. Programming language constructs for which it is impossible to obtain good Hoare axiom systems. *Journal of the ACM*, 26,1, pp. 129-147, January 1979.

2. S.A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM Journal of Computing*, Vol. 7, No. 1, pp. 70-90, February 1978.

3. E.W. Dijkstra. *A discipline of programming*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

4. C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, vol. 12, no. 10, pp. 576-580, October 1969.

5. C.A.R. Hoare and N. Wirth. An axiomatic definition of the programming language Pascal. *Acta Informatica*, 2, 4, pp. 335-355, 1973.

6. S. Igarashi, R.L. London, D.C. Luckham. Automatic program verification I: A logical basis and its implementation. *Acta Informatica*, 4, pp. 145-182, 1975.

7. R.L. London, J.V. Guttag, J.J. Horning, B.W. Lampson, J.G. Mitchell, and G.J. Popek. Proof rules for the programming language Euclid. *Acta Informatica*, 10, pp. 1-26, 1978.

8. A. Meyer and J. Halpern. Axiomatic definitions of programming languages: a theoretical assessment. *Seventh Annual ACM Symposium on Principles of Programming Languages*, pp. 203-212, January 1980.

9. R. Schwartz. An axiomatic semantic definition of Algol 68. Ph.D. thesis, UCLA Computer Science Department Report UCLA-34-P214-75, August 1978.

10. R. Schwartz. An Axiomatic Treatment of Algol 68 Routines. *Proceedings of the International Conference on Automata, Languages and Programming*, Springer Verlag Lecture Notes in Computer Science, July 1979.

Appendix: Equivalence-Preserving Transformation to Normal Form

The transformation from the general rule form of Section 4 to the more constrained normal form of Section 3 is defined as follows. First sort the rule according to the three classes of allowable premises, yielding a schema of the form

$$\frac{\mathcal{F}_1\{S_1\}\mathcal{Q}_1 ,\ldots, \mathcal{F}_j\{S_j\}\mathcal{Q}_j , \mathcal{R}_{j+1}\{S_{j+1}\}\mathcal{Q}_{j+1} ,\ldots, \mathcal{R}_k\{S_k\}\mathcal{Q}_k , \quad \mathcal{R}_{k+1}\wedge\mathcal{F}_{k+1}\{S_{k+1}\}\mathcal{Q}_{k+1} ,\ldots, \mathcal{R}_n\wedge\mathcal{F}_n\{S_n\}\mathcal{Q}_n, \Gamma}{\mathcal{P}\{S\}\, Q}$$

We now define two functions:

$$\text{Duplicates(i)} \;=\; \{m : |\mathcal{R}_m| = |\mathcal{R}_i|,\ j+1\leq m\leq n\}, \quad \text{for } j+1\leq i\leq n$$

where, for a metavariable $\mathcal{R}$, $|\mathcal{R}|$ denotes the partially interpreted first-order formula bound to $\mathcal{R}$, and

$$\text{MkFormula(i)} \;=\; P_i \text{ (for } j+1\leq i\leq k) \text{ and } |\mathcal{F}_i|\supset P_i \text{ (for } k+1\leq i\leq n)$$

Now rewrite the sorted schema above as

$$\frac{P_1\{S_1\}\mathcal{Q}_1, \ldots, P_n\{S_n\}\mathcal{Q}_n, \Gamma \wedge (|\mathcal{F}_1|\supset P_1) \wedge\ldots\wedge (|\mathcal{F}_j|\supset P_j)}{\mathcal{P}\{S\}\, Q}$$

with the subsequent overall proper substitution

$$[\,|\mathcal{R}_i| \leftarrow \bigwedge\nolimits_{k\in \text{Duplicates(i)}} \text{MkFormula(k)}], \quad \text{for } j+1\leq i\leq n$$

The last step is to reorder the premises of this rule to satisfy normal form constraint 4a, which can always be done.