Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P. America, F.S. de Boer

Proving total correctness of recursive procedures

Computer Science/Department of Software Technology Report CS-R8904 March

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Proving Total Correctness of Recursive Procedures

Pierre America
Philips Research Laboratories
P.O. Box 80000
5600 JA Eindhoven
The Netherlands

Frank de Boer
Centre for Mathematics and Computer Science
P.O. Box 4079
1009 AB Amsterdam
The Netherlands

March 15, 1989

Abstract

We show that some well-known rules in a Hoare-style proof system for total correctness of recursive procedures can interact in such a way that they yield incorrect results. The problem is connected to the quantification scope of certain variables in the proof rules. By defining some restrictions on the applicability of the rules a system is obtained that is sound and complete. However, the completeness proof differs substantially from the original one.

This technique is also applied to dynamic logic, where we show that the original proof rules for recursive procedures can be replaced by simpler and more natural ones, and that it is not necessary to extend the programming language in order to arrive at a sound and complete proof system.

Key words and Phrases: Proof theory, Recursion, Total correctness, Soundness, Completeness.
1980 Mathematics Subject Classification: 68B10.
1985 CR Categories: F.3.1.

Most of this work has been carried out in the context of ESPRIT project 415: "Parallel Architectures and Languages for Advanced Information Processing: A VLSI-directed Approach"

1 Introduction

A vast field of research in theoretical computer science is the formalisation of program correctness. This research has resulted in a variety of programming logics, of which we mention: *Hoare logic* [Ho], *dynamic logic* [Ha], and *temporal logic* [PM]. In Hoare logic, one of the formalisms that we shall use, a program is seen as state transformer: A state assigns a value to each program variable and a program transforms an initial state (before the execution) into the corresponding final state (after the execution). One specifies the input/output behaviour of programs by means of triples

$$\{p\}S\{q\}$$

where p and q are formulas of first-order predicate logic and S denotes a program. The formula p is called the *precondition* of S: it specifies a set of initial states. The corresponding set of final states is denoted by the formula q, which is called the *postcondition* of S.

There are two common ways to interpret these Hoare triples. One interpretation of $\{p\}S\{q\}$ is the following:

> If the execution of S in a state satisfying p terminates, it does so in a state satisfying q.

This gives rise to what is called *partial correctness*. On the other hand, *total correctness* uses the following interpretation:

> Every execution of S starting in a state satisfying p terminates in a state satisfying q.

Note that the total correctness interpretation additionally guarantees the termination of S when started in a state satisfying the precondition p.

The first subject of this paper is a Hoare-style logic to reason about the *total* correctness of programs. We shall be concerned with a *proof system*, i.e., a set of axioms and rules by which one can *derive* correctness formulas. (In this paper we shall use the term 'correctness formula' to refer to either a first-order formula or a Hoare triple as described above). For such a proof system two concepts are especially important: A proof system is called *sound* if every correctness formula that can be derived from it is indeed *valid*, i.e., if it really describes the behaviour of the corresponding program. (Of course, this should be measured against some formally defined semantics of the programming language.) On the other hand, a proof system is called *complete* if it can derive any valid correctness formula.

The programming language we consider will contain recursive, parameterless procedures. The basic statements of this language are assignments and procedure calls. Complex statements are constructed from these basic ones by sequential composition, conditional, and the while construct.

In [S], Sokołowski presented a rule for proving the total correctness of recursive procedures. Apt [A] however proved that this rule does not enable one to derive

all valid correctness formulas. In addition one needs some rules which formalise the reasoning about certain invariance properties of procedure calls, properties stating that the initial value of a variable that is not used in the procedure equals its final value (the value after the execution of the particular procedure). The resulting proof system, presented in [A], however turns out to be *unsound*, that is, one can derive from it correctness formulas which are not valid. The unsoundness of the system is due to the interaction of the recursion rule, the rule which enables one to reason about procedure calls, on the one hand, and those rules which formalise the invariance properties of these calls on the other hand. It turns out that the problem is due to the fact that the two sets of rules need different interpretations with respect to the scope of the implicit quantification applied to free variables.

We will formulate some restrictions on the applicability of those rules which can interact in an incorrect way, and prove that the resulting system is sound. Furthermore we will prove that even with these restrictions the resulting proof system is still complete. The proof of the completeness theorem differs from the one given in [A] because in the proof given there, our restrictions are not satisfied. As the proof in [A] of the completeness theorem for total correctness follows the same pattern as the one for the proof system for the partial correctness for the same programming language, we may conclude that reasoning about total correctness differs from partial correctness in a substantial way which has not been recognised till now.

After that we show that the techniques mentioned above can also be applied fruitfully to dynamic logic [Ha], another formalism to reason about the correctness of programs. In this logic the quantification scope can be mentioned explicitly. With our techniques, it is possible to give simpler and more natural rules for recursive procedures than the ones given in [Ha]. In particular, in our system it is not necessary to artificially extend the programming language.

Our paper is organised as follows: In the following section we present the programming language and define its semantics. In the third section we give the proof system as presented in [A], and analyse its unsoundness. Then, in section 4, we formulate some appropriate restrictions on the applicability of those rules which may interact in an incorrect way, and prove that these restrictions give rise to a sound proof system. The completeness of this new system is proved in section 5. In section 6 we apply our technique to dynamic logic and section 7 presents some conclusions.

2 The programming language

In this section we present the programming language which is the subject of our study. We shall give a formal definition of its semantics. We conclude this section by formally defining the total correctness interpretation of correctness formulas.

2.1 Syntax

We fix a set L of function and predicate symbols (and containing the equality symbol), a set Var of variables, typical elements of which are denoted by x, y, z, ..., and a

set *Proc* of procedure identifiers with typical element P. A *term* of L is a construct built up from variables and the function symbols of L. Terms are denoted by $t, \ldots$. A *boolean expression* of L is a construct built up from the terms, predicate symbols of L, and the usual logical connectives like $\wedge$, $\vee$, $\rightarrow$, and $\neg$. Boolean expressions are denoted by $b, \ldots$.

First-order predicate logic formulas of L are denoted by $p, q, \ldots$. By $FV(p)$ we denote the set of variables occurring free in the formula p. By $p(x_1, \ldots, x_n)$ we mean a formula p such that $\{x_1, \ldots, x_n\} \subseteq FV(p)$. A sequence of variables $x_1, \ldots, x_n$ will sometimes be written as $\overline{x}$. Equally we shall sometimes denote a sequence $t_1, \ldots, t_n$ of terms by $\overline{t}$. Now $p[\overline{t}/\overline{x}]$ will denote the result of the simultaneous substitution of t_i for the free occurrences of x_i, assuming that the x_i are distinct. When it is clear from the context which variables are substituted for, we sometimes abbreviate $p[\overline{t}/\overline{x}]$ to $p(\overline{t})$. We shall denote syntactic equality by the symbol '$\equiv$'.

We define the class of statements by means of the following grammar:

$$S ::= P \mid x := t \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi} \mid \textbf{while } b \textbf{ do } S \textbf{ od}$$

By $Var(S)$ we denote the set of variables occurring explicitly in S.

Programs are of the following form:

$$\langle P_1 \leftarrow S_1, \ldots, P_n \leftarrow S_n \mid S \rangle$$

where all the P_i are distinct, and only the procedure identifiers $P_1, \ldots, P_n$ occur in $S_1, \ldots, S_n$ and S. The first part of a program consists of declarations, associating with each procedure identifier P_i a statement S_i, which is called the *body* of the procedure P_i. The second part of a program is called its *initial statement*. Execution of the program means execution of its initial statement in the context established by the declarations. Note that occurrences of P_i in S_j, $1 \leq i, j \leq n$, give rise to the phenomenon of (mutual) *recursion*.

Just for the sake of convenience we shall restrict ourselves in this paper to programs of the form $\langle P \leftarrow S_0 \mid S \rangle$. Furthermore we shall drop the declaration $P \leftarrow S_0$ and just write S, assuming the declaration $P \leftarrow S_0$ to be fixed. It is a straightforward, but tedious task to generalise the results of this paper to programs with more than one procedure.

2.2 Semantics

An *interpretation* I of L consists of a set I_D, which is called I's domain of values, and a mapping which associates an operation on I_D with each function symbol of L and a relation on I_D with each predicate symbol of L. Throughout this paper we shall mostly assume a fixed interpretation I. We denote the elements of I_D by $d, \ldots$.

We define the set $\Sigma(I)$ of *states* (over I) by

$$\Sigma(I) = Var \rightarrow I_D.$$

Typical elements of $\Sigma(I)$ are denoted by $\sigma, \ldots$.

By $\Sigma(I)_\perp$ we denote the set $\Sigma(I)$ extended with some new element $\perp$ (pronounced "bottom"). We assume the following partial ordering on $\Sigma(I)_\perp$: For $\sigma_1, \sigma_2 \in \Sigma(I)_\perp$ we put

$$\sigma_1 \leq \sigma_2 \qquad \text{iff} \qquad \sigma_1 = \perp \text{ or } \sigma_1 = \sigma_2.$$

This ordering turns $\Sigma(I)_\perp$ into a complete partial order. The least upper bound of a sequence $(\sigma_n)_n$ of elements of $\Sigma(I)_\perp$ such that $\sigma_n \leq \sigma_{n+1}$ will be denoted by $\bigsqcup_n \sigma_n$.

Given $\sigma \in \Sigma(I)$ and a term t, $\sigma(t)$ will denote the result of evaluating t in the state σ (so $\sigma(t) \in I_D$). For a sequence $\overline{t}$ of terms, $\sigma(\overline{t})$ denotes the sequence of values $\sigma(t_1), \ldots, \sigma(t_n)$.

Given a first-order formula p in L and a state $\sigma \in \Sigma(I)$, the truth of p in σ with respect to the interpretation I, denoted by $\sigma \models_I p$, is defined as usual. We shall write $\models_I p$ if $\sigma \models_I p$ for every $\sigma \in \Sigma(I)$.

Let $\sigma \in \Sigma(I)$, $\overline{d}$ a sequence of elements of I_D, and $\overline{x}$ a sequence (of the same length) of distinct variables, then we define $\sigma\{\overline{d}/\overline{x}\} \in \Sigma(I)$ such that

$$\sigma\{\overline{d}/\overline{x}\}(y) = \begin{cases} d_i & \text{if } y \equiv x_i \\ \sigma(y) & \text{otherwise} \end{cases}$$

To construct a semantics of the programming language as defined in the previous section it is convenient to extend the language by the following statements: **skip**, execution of which consists of doing nothing, and Ω, execution of which never terminates.

Definition 2.1
We define for an arbitrary program S the semantic function

$$M_I(S) : \Sigma(I)_\perp \to \Sigma(I)_\perp,$$

transforming any initial state to the corresponding final state, as follows (we assume the declaration $P \leftarrow S_0$):

- $M_I(S)(\perp) = \perp$ for arbitrary S. Assume from here on that $\sigma \neq \perp$.
- $M_I(x := t)(\sigma) = \sigma\{\sigma(t)/x\}$
- $M_I(\textbf{skip})(\sigma) = \sigma$
- $M_I(\Omega)(\sigma) = \perp$
- $M_I(P)(\sigma) = \bigsqcup_k M_I(S_0^{(k)})(\sigma)$
  where $S_0^{(0)} \equiv \Omega$
  $S_0^{(k+1)} \equiv S_0[S_0^{(k)}/P]$
  and where $S[S'/P]$ denotes the result of replacing every occurrence of P in S by S'.
- $M_I(S_1; S_2)(\sigma) = M_I(S_2)(M_I(S_1)(\sigma))$

- $M_I(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi})(\sigma) = \begin{cases} M_I(S_1)(\sigma) & \text{if } \sigma \models_I b \\ M_I(S_2)(\sigma) & \text{otherwise} \end{cases}$
- $M_I(\textbf{while } b \textbf{ do } S \textbf{ od})(\sigma) = \bigsqcup_k M_I(S^{\{k\}})(\sigma)$
  where $S^{\{0\}} \equiv \textbf{if } b \textbf{ then } \Omega \textbf{ else skip fi}$
  $S^{\{k+1\}} \equiv \textbf{if } b \textbf{ then } S; S^{\{k\}} \textbf{ else skip fi}$

This definition is inductive on the complexity of statements, measured as follows: We define $C(S)$ to be a triple of natural numbers, and we order these triples lexicographically. The first component of $C(S)$, denoted by $C_1(S)$, is 1 if P occurs in S, 0 otherwise. The second component of $C(S)$, denoted by $C_2(S)$, gives the maximal nesting level of while statements occurring in S, and the third component of $C(S)$ gives the length of S. Now for the semantic definition of a procedure call we have a decrease in the first component, since $C_1(P) = 1$ while $C_1(S_0^{(k)}) = 0$. For the semantic definitions of the sequential composition and the conditional we have a decrease in the third component, while the first two components do not increase. Finally, for the while statement we have a decrease in the second component, while the first one does not increase.

The well-definedness of this semantics follows from the following propositions:

Proposition 2.2
For any natural number $k \in \mathbf{N}$, for any $\sigma \in \Sigma(I)$, and for any statement S we have

$$M_I(S^{\{k\}})(\sigma) \sqsubseteq M_I(S^{\{k+1\}})(\sigma),$$

where we assume $S^{\{k\}}$ to be defined with respect to some fixed boolean expression b.

Proof
Induction on k. □

Proposition 2.3
For arbitrary statements S, S_1, and S_2 such that $M_I(S_1)(\sigma) \sqsubseteq M_I(S_2)(\sigma)$ for every $\sigma \in \Sigma(I)$, we have

$$M_I(S[S_1/P])(\sigma) \sqsubseteq M_I(S[S_2/P])(\sigma)$$

for every $\sigma \in \Sigma(I)$.

Proof
Induction on the complexity of S. □

Proposition 2.4
For all $k \in \mathbf{N}$ and for all $\sigma \in \Sigma(I)$ we have

$$M_I(S_0^{(k)})(\sigma) \sqsubseteq M_I(S_0^{(k+1)})(\sigma).$$

Proof
Induction on k, using proposition 2.3. □

We conclude this subsection with some propositions that will be used in the proofs of the soundness and the completeness theorem.

Proposition 2.5
For an arbitrary interpretation I, a state $\sigma \in \Sigma(I)$, a sequence $\bar{t}$ of terms, and a sequence $\bar{x}$ of distinct variables, we have

$$\sigma \models_I p[\bar{t}/\bar{x}] \qquad \text{iff} \qquad \sigma\{\sigma(\bar{t})/\bar{x}\} \models_I p.$$

Proof
Induction on the complexity of p. □

Proposition 2.6
Let I be an arbitrary interpretation, p a formula, and let $\sigma, \sigma' \in \Sigma(I)$ such that σ agrees with σ' on the variables occurring free in p. Then

$$\sigma \models_I p \qquad \text{iff} \qquad \sigma' \models_I p.$$

Proof
Induction on the complexity of the formula p. □

Proposition 2.7
Let I be an interpretation, S a program, $\bar{y}$ a sequence of variables such that $\bar{y} \cap Var(S, S_0) = \emptyset$, and $\bar{d}$ a sequence of data in I_D. Let $\sigma \in \Sigma(I)$, put $\sigma' = M_I(S)(\sigma)$, and suppose $\sigma' \neq \perp$. Then

$$\sigma'\{\bar{d}/\bar{y}\} = M_I(S)(\sigma\{\bar{d}/\bar{y}\}).$$

Informally speaking, this means that a program S only depends on and accesses the variables that occur explicitly in S or in S_0.

Proof
Induction on the complexity of S. □

Lemma 2.8
For any statement S and any state $\sigma \in \Sigma(I)$ we have

$$M_I(S)(\sigma) = \bigsqcup_k M_I(S[S_0^{(k)}/P])(\sigma).$$

Proof
Induction on the complexity of S. The following property of the ordering on $\Sigma(I)_\perp$ is heavily used: If $\sigma = \bigsqcup_k \sigma_k$, where the σ_k form a nondecreasing sequence, then there exists a k_0 such that $\sigma = \sigma_k$ for all $k \geq k_0$. □

Proposition 2.9
For any state $\sigma \in \Sigma(I)$ we have

$$M_I(S_0)(\sigma) = M_I(P)(\sigma).$$

Proof
Applying lemma 2.8 to S_0 we get

$$\begin{aligned} M_I(S_0)(\sigma) &= \bigsqcup_k M_I(S_0[S_0^{(k)}/P])(\sigma) \\ &= \bigsqcup_k M_I(S_0^{(k+1)})(\sigma) \\ &= \bigsqcup_k M_I(S_0^{(k)})(\sigma) \\ &= M_I(P)(\sigma). \end{aligned}$$

□

2.3 Total correctness

Correctness formulas are triples of the form $\{p\}S\{q\}$, where p and q are first-order formulas of L. We shall interpret such correctness formulas with respect to so-called *arithmetical* interpretations [Ha], as defined in

Definition 2.10
Let L^+ be the set of symbols of L extended with some one-place predicate '*nat*' and with the usual function and relation symbols for describing the arithmetic of the natural numbers (like addition, multiplication and comparison). An interpretation I of L^+ is called *arithmetical* if the following conditions are satisfied:

- I_D contains the standard model of Peano arithmetic, that is, (a copy of) the set $\mathbf{N}$ of natural numbers.
- The predicate *nat* is interpreted in such a way that for arbitrary $\sigma \in \Sigma(I)$

$$\sigma \models_I nat(x) \qquad \text{iff} \qquad \sigma(x) \in \mathbf{N}.$$

- The arithmetical function and relation symbols in L^+ are given the standard interpretation, i.e., they are mapped to the standard operations on the natural numbers in I_D.
- There exists a formula of L^+ defining some coding of the finite sequences of I_D. More precisely: There exists an injective mapping f from the set of finite sequences of elements of I_D to I_D itself, and a formula $\phi(x, y, n)$ which represents this mapping in the following sense: For arbitrary $\sigma \in \Sigma(I)$ we have $\sigma \models_I \phi(x, y, n)$ iff, for some sequence $d_1, \ldots, d_k$ in I_D, $f(\langle d_1, \ldots, d_k \rangle) = \sigma(x)$, $1 \leq \sigma(n) \leq k$, and $\sigma(y) = d_{\sigma(n)}$.

Arithmetical interpretations are important for the following reasons: Firstly, the basic pattern in reasoning about termination involves a well-foundedness argument, that is, with the body of a procedure or while statement we associate a formula $p(x)$, where the variable x ranges over some set on which some well-founded ordering is

defined. Termination of the particular procedure or while statement is then proved by showing that $p(x)$ holds initially, and that $p(x)$ holds after every execution of the corresponding body for some smaller value of x. It is always possible to take the set of natural numbers with the standard ordering for this purpose, and by requiring that this set is contained in the interpretation I, we can carry out the above termination argument in a formal way.

The second reason is that to prove completeness the interpretations we consider must have sufficient expressive power to represent the notion of an execution in our assertions. In order to do this, we must be able to reason about natural numbers and about sequences of data elements.

In any case, it is shown in [A] that even for a language without recursive procedures (but with while statements) there exists no adequate proof system for total correctness that is sound for arbitrary (nonarithmetical) interpretations.

Now we define the truth, or validity of a correctness formula $\{p\}S\{q\}$ with respect to some arbitrary interpretation I:

Definition 2.11
Let I be an arbitrary interpretation. For $\sigma \in \Sigma(I)$ we write $\sigma \models_I \{p\}S\{q\}$ iff $\sigma \models_I p$ implies that there exists a $\sigma' \neq \perp$ such that $\sigma' = M_I(S)(\sigma)$ and $\sigma' \models_I q$. We write $\models_I \{p\}S\{q\}$ iff for all $\sigma \in \Sigma(I)$ we have $\sigma \models_I \{p\}S\{q\}$.

Although we have defined $\models_I$ for any arbitrary interpretation I, we shall be interested in it only if I is arithmetical.

3 The proof system G_0

In this section we give the proof system G_0 as presented in [A]. We give two examples of derivations in this proof system, of which one leads to a valid correctness formula and the other to an invalid one.

Definition 3.1
The proof system G_0 consists of the following axioms and rules:

Assignment:
$$\{p[t/x]\}\ x := t\ \{p\}$$

Sequential composition:
$$\frac{\{p\}S_1\{r\} \quad \{r\}S_2\{q\}}{\{p\}S_1; S_2\{q\}}$$

Conditional:
$$\frac{\{p \wedge b\}S_1\{q\} \quad \{p \wedge \neg b\}S_2\{q\}}{\{p\}\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}\{q\}}$$

Iteration:
$$\frac{\{p(m+1)\}S\{p(m)\} \quad p(m+1) \rightarrow b \quad p(0) \rightarrow \neg b}{\{\exists m\, p(m)\}\textbf{while } b \textbf{ do } S \textbf{ od}\{p(0)\}}$$

provided $m \notin Var(S) \cup Var(S_0)$.

Recursion:
$$\frac{\neg p(0) \quad \{p(n)\}P\{q\} \vdash \{p(n+1)\}S_0\{q\}}{\{\exists n\, p(n)\}P\{q\}}$$

provided $n \notin FV(q) \cup Var(S_0)$.

Invariance:
$$\frac{\{p\}P\{q\}}{\{p \wedge r\}P\{q \wedge r\}}$$

provided $FV(r) \cap Var(S_0) = \emptyset$.

Elimination:
$$\frac{\{p\}P\{q\}}{\{\exists z\, p\}P\{q\}}$$

provided $z \notin FV(q) \cup Var(S_0)$.

Substitution:
$$\frac{\{p\}P\{q\}}{\{p[y/z]\}P\{q[y/z]\}}$$

provided $y, z \notin Var(S_0)$.

Consequence:
$$\frac{p \to p_1 \quad \{p_1\}S\{q_1\} \quad q_1 \to q}{\{p\}S\{q\}}$$

The variables n and m occurring in the rules for recursion and iteration are supposed to range over the set of natural numbers, that is, $p(n)$, for example, is an abbreviation of $p(n) \wedge nat(n)$.

The premiss $\{p(n)\}P\{q\} \vdash \{p(n+1)\}S_0\{q\}$ of the recursion rule states that it is possible to derive $\{p(n+1)\}S_0\{q\}$ if one takes $\{p(n)\}P\{q\}$ as an *assumption*. The intuition behind the recursion rule is that the value of the variable n in a state satisfying $p(n)$ gives an upper bound to the number of nested calls necessary to complete the computation of the procedure P starting from this state.

The notion of derivability is defined relative to some interpretation:

Definition 3.2
Let I be an arithmetical interpretation. We write $\vdash_I \{p\}S\{q\}$ to denote that the correctness formula $\{p\}S\{q\}$ is derivable from the axioms and rules of the proof system, making use of the (first-order) theory of the interpretation I in the iteration, recursion, and consequence rules. (In other words, every assertion p such that $\models_I p$ can be used as an axiom.)

Let us give an example of a derivation which illustrates the use of the recursion, elimination, invariance, and the substitution rule. Consider the well-known recursive procedure which calculates the factorial of the number stored in the variable x:

$P \leftarrow$ **if** $x = 0$
then $y := 1$
else $x := x - 1; P; x := x + 1; y := y \times x$
fi

We shall prove that this procedure indeed calculates the factorial of x, and moreover that the value of x after the execution equals the initial value of x. For the latter purpose, we use the variable z, which does not occur in the program, as a "freeze variable". Since we know that the program does not change the value of z (see proposition 2.7), we can use z to remember the value of x in the initial state. Therefore we want to derive the correctness assertion

$$\{x = z \geq 0\}P\{x = z \wedge y = x!\}$$

We will do so by first proving that

$$\{x = z = n-1 \geq 0\}P\{x = z \wedge y = x!\} \vdash_I \{x = z = (n+1)-1 \geq 0\}S_0\{x = z \wedge y = x!\}$$

where S_0 is the body of the procedure P, as defined above. We take an arbitrary arithmetical interpretation I, of which we shall only use the valid formulas that deal with natural numbers. Let $S \equiv x := x - 1; P; x := x + 1; y := y \times x$ and reason as follows (within the proof system):

1. $\{x = z = n - 1 \geq 0\}P\{x = z \wedge y = x!\}$, our assumption.
2. $\{x = u = n - 1 \geq 0\}P\{x = u \wedge y = x!\}$, from 1 by the substitution rule.
3. $\{x = u = n - 1 \geq 0 \wedge u = z - 1\}P\{x = u \wedge y = x! \wedge u = z - 1\}$, from 2 by the invariance rule.
4. $\{x = u = z - 1 = n - 1 \geq 0\}P\{x = z - 1 \wedge y = x!\}$, from 3 by the consequence rule.
5. $\{\exists u(x = u = z-1 = n-1 \geq 0)\}P\{x = z-1 \wedge y = x!\}$, from 4 by the elimination rule.
6. $\{x = z - 1 = n - 1 \geq 0)\}P\{x + 1 = z \wedge y \times (x + 1) = (x + 1)!\}$, from 5 by the consequence rule.
7. $\{x - 1 = z - 1 = n - 1 \geq 0\}S\{x = z \wedge y = x!\}$, by applying three times the assignment axiom and the rule for sequential composition, using 6.
8. $\{x = z = (n+1)-1 \geq 0 \wedge \neg x = 0\}S\{x = z \wedge y = x!\}$, from 7 by the consequence rule.
9. $\{x = z = (n + 1) - 1 \geq 0 \wedge x = 0\}y := 1\{x = z \wedge y = x!\}$, by the assignment axiom and the rule of consequence.
10. $\{x = z = (n + 1) - 1 \geq 0\}S_0\{x = z \wedge y = x!\}$, by applying the rule for the conditional to 8 and 9.

In addition we have $\models_I \neg(x = z = 0 - 1 \geq 0)$, so applying the recursion rule yields

$$\{\exists n(x = z = n - 1 \geq 0)\}P\{x = z \wedge y = x!\}.$$

Now $\models_I x = z \geq 0 \rightarrow \exists n(x = z = n - 1 \geq 0)$, so applying the consequence rule gives us the desired result.

Note how we have used the rules for substitution, invariance, and elimination to change the context in which the procedure P is called. More precisely, we have renamed the freeze variable z, in order to call the procedure P for a different value of x.

The above derivation might give the impression that all works well. However, we shall now give an example of a derivation the conclusion of which is invalid (with respect to any interpretation), thus establishing the unsoundness of the system G_0. Consider the following declaration:

$$P \leftarrow P; P.$$

(The simplest example would be $P \leftarrow P$, but the above example is clearer because it has $P \not\equiv S_0$.) It is obvious that every computation of P diverges, so the correctness formula
$\{\mathbf{true}\}P\{\mathbf{true}\}$, stating that every computation of P terminates, is invalid. Nevertheless, the following derivation establishes the derivability of $\{\mathbf{true}\}P\{\mathbf{true}\}$:

1. $\{n > 0\}P\{\mathbf{true}\}$, assumption.
2. $\{\exists n(n > 0)\}P\{\mathbf{true}\}$, by the elimination rule.
3. $\{\mathbf{true}\}P\{\mathbf{true}\}$, by the consequence rule.
4. $\{\exists n(n > 0)\}P; P\{\mathbf{true}\}$, by applying the sequential composition rule to 2 and 3.
5. $\{n + 1 > 0\}P; P\{\mathbf{true}\}$, by the consequence rule.
6. $\{\exists n(n > 0)\}P\{\mathbf{true}\}$, by applying the recursion rule to 1–5 (Note that $\models_I \neg(0 > 0)$.)
7. $\{\mathbf{true}\}P\{\mathbf{true}\}$, by the consequence rule.

Note that in the first and last application of the consequence rule we made explicit use of the fact that the variable n ranges over the natural numbers, and that $>$ denotes the usual notion of "greater than".

To understand what went wrong in this derivation we shall investigate where exactly a formal justification breaks down. Establishing soundness consists of proving that, for every arithmetical interpretation I and for every correctness formula $\{p\}S\{q\}$, if $\vdash_I \{p\}S\{q\}$ then $\models_I \{p\}S\{q\}$.

Usually one proves the soundness of a proof system by first showing that the axioms are valid and then that the validity of the premisses of an arbitrary rule implies the validity of its conclusion. But with respect to the present proof system one runs into the difficulty that the recursion rule is really a metarule. To overcome this problem we follow the strategy presented in [A] of transforming the proof system

into an ordinary one, and reducing the problem of proving the soundness of the original system to proving the soundness of its transformed version.

We shall call the transformed system K. This system manipulates *correctness phrases* of the form $\Phi \rightarrow \Psi$, where Φ and Ψ are (possibly empty) sets of correctness formulas.

Definition 3.3
The proof system K is defined as follows:

- For each axiom ϕ of G_0, add the axiom $\Phi \rightarrow \phi$ to K.
- For each rule
$$\frac{\phi_1, \ldots, \phi_n}{\phi_{n+1}}$$
of G_0, except for the recursion rule, add the rule
$$\frac{\Phi \rightarrow \phi_1, \ldots, \Phi \rightarrow \phi_n}{\Phi \rightarrow \phi_{n+1}}$$
to K.

Finally add the following rules and axioms to K:

- $$\frac{\Phi \rightarrow \neg p(0) \quad \Phi, \{p(n)\}P\{q\} \rightarrow \{p(n+1)\}S_0\{q\}}{\Phi \rightarrow \{\exists n\, p(n)\}P\{q\}}$$
provided that n does not occur free in Φ, S_0, or q.
- $$\frac{\Phi \rightarrow \phi_1 \quad \ldots \quad \Phi \rightarrow \phi_n}{\Phi \rightarrow \phi_1, \ldots, \phi_n}$$
- $\Phi \rightarrow \phi$, for every $\phi \in \Phi$.

Let $\vdash_I^K \Phi \rightarrow \Psi$ denote the derivability of the correctness phrase $\Phi \rightarrow \Psi$ in K making use of the additional axioms $\Phi' \rightarrow p$, for any first-order formula p such that $\models_I p$. $\vdash_I^{G_0} \phi$ will now denote the derivability of the correctness formula ϕ in G_0 making use of the first-order theory of I as additional axioms. We shall write $\Phi \vdash_I^{G_0} \phi$ to denote that ϕ can be derived in the proof system G_0 using as axioms the elements of Φ (in addition to the first-order theory of I).

Lemma 3.4
For any arithmetical interpretation I and any correctness phrase of the form $\Phi \rightarrow \phi$, where ϕ is a (single) correctness formula, we have

$$\Phi \vdash_I^{G_0} \phi \qquad \text{implies} \qquad \vdash_I^K \Phi \rightarrow \phi.$$

Proof
A simple induction of the length of a derivation for $\Phi \vdash_I^{G_0} \phi$. □

It follows that for any arithmetical interpretation I and any correctness formula ϕ

$$\vdash_I^{G_0} \phi \qquad \text{implies} \qquad \vdash_I^{K} \emptyset \to \phi.$$

Thus, given a definition of validity for correctness phrases which agrees on correctness formulas with definition 2.11, the soundness of K implies the soundness of G_0.

We next look at two ways to interpret the correctness phrases. In either way, however, the system K will turn out to be unsound.

First consider the following way to interpret correctness phrases: For an arbitrary interpretation I we define

$$\models_I \Phi \to \Psi \qquad \text{iff} \qquad \models_I \Phi \text{ implies } \models_I \Psi$$

where $\models_I \Phi$ iff $\models_I \phi$ for all $\phi \in \Phi$.

This definition, however, will make the recursion rule unsound: Consider the declaration $P \leftarrow P;P$ and take $p(n) \equiv n > 0$, $q \equiv \mathbf{true}$. Take some arbitrary arithmetical interpretation I, then $\models_I \neg(0 > 0)$ and

$$\models_I \{p(n)\}P\{q\} \to \{p(n+1)\}S_0\{q\},$$

because it is not the case that $\models_I \{p(n)\}P\{q\}$: For any state $\sigma \in \Sigma(I)$ such that $\sigma(n) > 0$ we have $\sigma \models_I p(n)$, but nevertheless $M_I(P)(\sigma) = \bot$. Therefore the premisses of the recursion rule are valid. However, it is not the case that $\models_I \{\exists n\, p(n)\}P\{q\}$ (note that $\sigma \models_I \exists n\, p(n)$ for every $\sigma \in \Sigma(I)$).

The unsoundness of the recursion rule with respect to this interpretation is due to the fact that the variable n is universally quantified at both sides of the implication sign independently, so that it does not retain its value over the implication.

This suggests that one should define the validity of a correctness phrase as follows:

$$\models_I \Phi \to \Psi \qquad \text{iff} \qquad \sigma \models_I \Phi \text{ implies } \sigma \models_I \Psi \text{ for all } \sigma \in \Sigma(I)$$

where we define $\sigma \models_I \Phi$ iff $\sigma \models_I \phi$ for all $\phi \in \Phi$.

This second way to interpret correctness phrases will, however, make the elimination rule unsound: Take the same declaration for P as above, then for any arithmetical interpretation I we obviously have

$$\models_I \{n > 0\}P\{\mathbf{true}\} \to \{n > 0\}P\{\mathbf{true}\},$$

but it is not the case that

$$\models_I \{n > 0\}P\{\mathbf{true}\} \to \{\exists n(n > 0)\}P\{\mathbf{true}\}.$$

To see this take a $\sigma \in \Sigma(I)$ with $\sigma(n) = 0$. Then $\sigma \models_I \{n > 0\}P\{\mathbf{true}\}$, because $\sigma \not\models_I n > 0$, but $\sigma \not\models_I \{\exists n(n > 0)\}P\{\mathbf{true}\}$ because $\sigma \models_I \exists n(n > 0)$ and P does not terminate.

This analysis suggests the following solution: Introduce a set *Count* of variables ranging over natural numbers. Interpret such variables as described in the second

case above, i.e. they are interpreted as being universally quantified, the scope of the quantification being the correctness phrase in which they occur. The other variables are interpreted as being universally quantified at both sides of the implication sign independently. We shall not allow variables of this set *Count* to occur in programs, to be quantified over by the elimination rule, nor to be substituted for in the substitution rule. On the other hand, only variables of the set *Count* are to be used in the recursion rule to establish termination of the particular procedure. This solution we shall work out in the following section.

4 The proof system T

The new proof system is defined by adding some restrictions on the applicability of some rules of the system G_0, as described in the previous section.

Let *Count* be a set of variables ranging over the natural numbers. Variables of this set will be called counter variables. We do not allow counter variables to occur in programs.

Definition 4.1
The proof system T consists of the same axioms and rules as the proof system G_0, but for the following restrictions:

Iteration rule: The variable m occurring in the rule, used to establish termination of the iteration construct, may not be a counter variable.

Recursion rule: The variable n used to establish termination of the procedure P must be a counter variable.

Elimination rule: The quantified variable z may not be a counter variable.

Substitution rule: Let y and z be the variables such that y is substituted for z in the conclusion of the rule. Then we require that $y, z \notin Count$.

Note that the derivation given in the previous section to establish the unsoundness of the system G_0 is not a correct derivation in the system K, because the variable n must be an element of *Count* for the recursion rule to be applicable, but then application of the elimination rule is not allowed.

4.1 Soundness

In this subsection we prove the soundness of the system T. Let K' be the proof system which manipulates correctness phrases as described in section 3, but now generated from T instead of G_0. To be able to reduce the problem of proving the soundness of T to that of K' we first have to define the notion of validity for correctness phrases.

Definition 4.2
Let Φ be a set of correctness formulas such that no counter variable occurs free in ϕ, for any $\phi \in \Phi$, and let I be an arithmetical interpretation. We define

$$\models_I \Phi \qquad \text{iff} \qquad \models_I \phi \text{ for all } \phi \in \Phi.$$

Definition 4.3
For any natural number $k \in \mathbf{N}$, let $\underline{k}$ be a constant term in the first-order language denoting k, that is, $\sigma(\underline{k}) = k$ for any state σ. Let $n_1, \ldots, n_l$ be all the counter variables occurring free in the correctness phrase $\Phi \to \Psi$. If $k_1, \ldots, k_l \in \mathbf{N}$, then by $[\underline{k_1}, \ldots, \underline{k_l}/n_1, \ldots, n_l]$, abbreviated to $[\overline{\underline{k}}/\overline{n}]$, we denote the simultaneous substitution of $\underline{k_i}$ for n_i. More precisely, with $(\{p\}S\{q\})[\overline{\underline{k}}/\overline{n}]$ we denote $\{p[\overline{\underline{k}}/\overline{n}]\}S\{q[\overline{\underline{k}}/\overline{n}]\}$, and with $\Phi[\overline{\underline{k}}/\overline{n}]$ we denote $\{\phi[\overline{\underline{k}}/\overline{n}] \mid \phi \in \Phi\}$. Now for any arithmetical interpretation I we define

$$\models_I \Phi \to \Psi \qquad \text{iff} \qquad \text{for all } \overline{k} \text{ in } \mathbf{N} \quad \models_I \Phi[\overline{\underline{k}}/\overline{n}] \quad \text{implies} \quad \models_I \Psi[\overline{\underline{k}}/\overline{n}].$$

Note that the notation $\models_I$ is used to denote both the truth of a correctness formula and that of a correctness phrase. The following proposition however states that this interpretation of correctness phrases agrees on correctness formulas with the interpretation as given in definition 2.11:

Proposition 4.4
For any arithmetical interpretation I and for any correctness formula $\{p\}S\{q\}$ we have

$$\models_I \{p\}S\{q\} \qquad \text{iff} \qquad \text{for all } \overline{k} \text{ in } \mathbf{N} \quad \models_I \{p[\overline{\underline{k}}/\overline{n}]\}S\{q[\overline{\underline{k}}/\overline{n}]\}$$

where $\overline{n}$ consists of all the counter variables occurring free in $\{p\}S\{q\}$.

Proof
$\Rightarrow$:
Let $\sigma \models_I p[\overline{\underline{k}}/\overline{n}]$, then by proposition 2.5 we have $\sigma_1 \models_I p$, where $\sigma_1 = \sigma\{\overline{k}/\overline{n}\}$. Now if $\sigma_1' = M_I(S)(\sigma_1)$, then $\sigma_1' \models_I q$, because $\models_I \{p\}S\{q\}$. On the other hand, if we put $\sigma' = M_I(S)(\sigma)$, then by proposition 2.7 we get

$$\sigma_1' = M_I(S)(\sigma_1) = M_I(S)(\sigma\{\overline{k}/\overline{n}\}) = \sigma'\{\overline{k}/\overline{n}\}$$

(note that $\overline{n} \cap Var(S) = \emptyset$). Applying proposition 2.5 again, we get $\sigma' \models_I q[\overline{\underline{k}}/\overline{n}]$.

$\Leftarrow$:
Let $\sigma \models_I p$ and let furthermore $\sigma(n_i) = k_i$ for $i = 1, \ldots, l$. Then by proposition 2.5 we have $\sigma \models_I p[\overline{\underline{k}}/\overline{n}]$, because $\sigma(\underline{k_i}) = \sigma(n_i)$, so that $\sigma\{\sigma(\overline{\underline{k}})/\overline{n}\} = \sigma$. So for $\sigma' = M_I(S)(\sigma)$ we get $\sigma' \models_I q[\overline{\underline{k}}/\overline{n}]$. Because $n_i \notin Var(S)$ we have $\sigma'(n_i) = \sigma(n_i) = k_i$ (this is implicit in proposition 2.7). Therefore we can apply proposition 2.5 again to get $\sigma' \models_I q$. □

We prove the soundness of the recursion and the elimination rule in separate lemmas:

Lemma 4.5 (Soundness of recursion rule)
Let I be an arithmetical interpretation and Φ a set of correctness formulas. Suppose that $\models_I \Phi \rightarrow \neg p(0)$ and

$$\models_I \Phi, \{p(n)\}P\{q\} \rightarrow \{p(n+1)\}S_0\{q\}.$$

Then

$$\models_I \Phi \rightarrow \{\exists n\, p(n)\}P\{q\},$$

provided $n \notin FV(\Phi, q)$.

Proof
Let $n_1, \ldots, n_l$ be all the counter variables occurring free in $\Phi \rightarrow \{\exists n\, p(n)\}P\{q\}$. Let $k_1, \ldots, k_l \in \mathbf{N}$ such that $\models_I \Phi[\overline{k}/\overline{n}]$. It is not difficult to see that $\sigma \models_I (\exists n\, p(n))[\overline{k}/\overline{n}]$ iff there is a $m \in \mathbf{N}$ so that $\sigma \models_I p(\underline{m})[\overline{k}/\overline{n}]$ (note that this only holds in ***arithmetical*** interpretations). From this it follows that

$$\models_I \{(\exists n\, p(n))[\overline{k}/\overline{n}]\}P\{q[\overline{k}/\overline{n}]\} \quad \text{iff} \quad \text{for all } m \in \mathbf{N} \quad \models_I \{p(\underline{m})[\overline{k}/\overline{n}]\}P\{q[\overline{k}/\overline{n}]\}.$$

We now prove that for all $m \in \mathbf{N}$

$$\models_I \{p(\underline{m})[\overline{k}/\overline{n}]\}P\{q[\overline{k}/\overline{n}]\}$$

by induction on m:

$m = 0$: By the hypothesis $\models_I \Phi \rightarrow \neg p(\underline{0})$ we have $\sigma \models_I \neg p(\underline{0})[\overline{k}/\overline{n}]$ for every $\sigma \in \Sigma(I)$, so

$$\models_I \{p(\underline{0})[\overline{k}/\overline{n}]\}P\{q[\overline{k}/\overline{n}]\}.$$

$m > 0$: We know that $n \notin FV(\Phi)$ so that $n \notin \{n_1, \ldots, n_l\}$ and $\models_I \Phi[\overline{k}/\overline{n}][\underline{m-1}/n]$. Therefore, from $\models_I \Phi, \{p(n)\}P\{q\} \rightarrow \{p(n+1)\}S_0\{q\}$ it follows that

$$\models_I \{p(n)[\overline{k}/\overline{n}][\underline{m-1}/n]\}P\{q[\overline{k}/\overline{n}][\underline{m-1}/n]\}$$
$$\text{implies}$$
$$\models_I \{p(n+1)[\overline{k}/\overline{n}][\underline{m-1}/n]\}S_0\{q[\overline{k}/\overline{n}][\underline{m-1}/n]\}.$$

Now $p(n)[\overline{k}/\overline{n}][\underline{m-1}/n] \equiv p(\underline{m-1})[\overline{k}/\overline{n}]$, and $p(n+1)[\overline{k}/\overline{n}][\underline{m-1}/n] \equiv p(\underline{m-1+1})[\overline{k}/\overline{n}]$. Furthermore $p(\underline{m-1+1})[\overline{k}/\overline{n}]$ is obviously semantically equivalent with $p(\underline{m})[\overline{k}/\overline{n}]$. Finally, because $n \notin FV(q)$ we have that $q[\overline{k}/\overline{n}][\underline{m-1}/n] \equiv q[\overline{k}/\overline{n}]$. So we get

$$\models_I \{p(\underline{m-1})[\overline{k}/\overline{n}]\}P\{q[\overline{k}/\overline{n}]\} \quad \text{implies} \quad \models_I \{p(\underline{m})[\overline{k}/\overline{n}]\}S_0\{q[\overline{k}/\overline{n}]\}.$$

Here the antecedent is just the induction hypothesis and the consequent is equivalent to

$$\models_I \{p(\underline{m})[\overline{k}/\overline{n}]\}P\{q[\overline{k}/\overline{n}]\}$$

by proposition 2.9. □

Lemma 4.6 (Soundness of elimination rule)
Let, for some arithmetical interpretation I and some set of correctness formulas Φ,

$$\models_I \Phi \to \{p\}P\{q\}.$$

Then

$$\models_I \Phi \to \{\exists z\, p\}P\{q\}$$

where $z \notin Count \cup Var(S_0) \cup FV(q)$.

Proof
Let $n_1, \ldots, n_l$ be all the counter variables occurring in the correctness phrase $\Phi \to \{\exists z\, p\}P\{q\}$. Let $k_1, \ldots, k_l \in \mathbf{N}$ be such that $\models_I \Phi[\underline{k}/\overline{n}]$. Let $\sigma \models_I (\exists z\, p)[\underline{k}/\overline{n}]$, so that for some $d \in I_D$ we have $\sigma\{d/z\} \models_I p[\underline{k}/\overline{n}]$. Then we know $\models_I \{p[\underline{k}/\overline{n}]\}P\{q[\underline{k}/\overline{n}]\}$, so for $\sigma' = M_I(P)(\sigma\{d/z\})$ we get $\sigma' \models_I q[\underline{k}/\overline{n}]$. Now $z \notin Var(S_0)$ implies $\sigma'\{\sigma(z)/z\} = M_I(P)(\sigma)$ by proposition 2.7. Finally, $z \notin FV(q)$ implies $\sigma'\{\sigma(z)/z\} \models_I q[\underline{k}/\overline{n}]\}$. We conclude:

$$\models_I \{\exists z\, p[\underline{k}/\overline{n}]\}P\{q[\underline{k}/\overline{n}]\}.$$

□

Lemma 4.7
For every arithmetical interpretation I we have

$$\vdash_I^{K'} \Phi \to \Psi \quad \text{implies} \quad \models_I \Phi \to \Psi.$$

Proof
The soundness of all the individual axioms and rules of K' can be shown along the lines of lemmas 4.5 and 4.6. The soundness of the whole proof system then follows by induction on the length of a derivation of $\vdash_I^{K'} \Phi \to \Psi$. □

Theorem 4.8
The proof system T is sound, that is, for every arithmetical interpretation I we have

$$\vdash_I^T \{p\}S\{q\} \quad \text{implies} \quad \models_I \{p\}S\{q\}.$$

Proof
This is now an easy consequence of (a slightly modified version of) lemma 3.4, lemma 4.7, and proposition 4.4. □

5 Completeness

In this section we prove the completeness of the system T, that is, we show that for any arithmetical interpretation I and any correctness formula $\{p\}S\{q\}$,

$$\models_I \{p\}S\{q\} \quad \text{implies} \quad \vdash_I \{p\}S\{q\}.$$

We assume the arithmetical interpretation I to be fixed throughout this section. To get started we need the following definitions and lemma:

Definition 5.1
For any program S and any natural number k, we define

$$S^{[k]} \equiv S[S_0^{(k)}/P].$$

(Remember that S_0 denotes the body of the procedure P and for $S_0^{(k)}$ recall definition 2.1.)

Lemma 5.2
For any program S, any first-order formula q, and any variable $n \in \mathit{Count}$ there exists a first-order formula $\mathit{Pre}(S, q, n)$, such that for any $\sigma \in \Sigma(I)$ we have $\sigma \models_I \mathit{Pre}(S, q, n)$ iff there exists a $\sigma' = M_I(S^{[k]})(\sigma) \neq \perp$ such that $\sigma' \models_I q$, where $k = \sigma(n)$.

Proof
The proof of this lemma is quite hard work. It consists of showing that the computation of $S^{[k]}$ can be coded in the first-order language. Here the fact that sequences of elements of I_D can be coded into single elements is essential. We shall not carry out the proof of this lemma here, but refer the reader to [TZ] and to the appendix of [B], where similar proofs are carried out in full detail. □

Definition 5.3
Take some variable $n \in \mathit{Count}$. We define

$$p_0(n) \equiv \mathit{Pre}(P, \overline{x} = \overline{z}, n),$$

where $\overline{x} = \mathit{Var}(S_0)$, $\overline{z} \cap \mathit{Count} = \emptyset$, and $\overline{z} \cap \mathit{Var}(S_0) = \emptyset$. (Here $\overline{x} = \overline{z}$ abbreviates the formula $x_1 = z_1 \wedge \cdots \wedge x_m = z_m$).

To appreciate the meaning of this definition, note that the formula $\exists n\, p_0(n)$ describes the 'graph' of the function $M_I(P)$ in the following sense: For arbitrary $\sigma \in \Sigma(I)$

$$\sigma \models_I \exists n\, p_0(n) \quad \text{implies} \quad \sigma\{\sigma(\overline{z})/\overline{x}\} = M_I(P)(\sigma)$$

and

$$\sigma' = M_I(P)(\sigma) \quad \text{implies} \quad \sigma\{\sigma'(\overline{x})/\overline{z}\} \models_I \exists n\, p_0(n).$$

Let us now outline the structure of the completeness proof. To do that we first describe the global structure of the proof given in [A]. There it is shown that for any valid correctness formula $\{p\}S\{q\}$

$$\{p_0(n)\}P\{\overline{x} = \overline{z}\} \vdash_I \{p\}S\{q\}$$

by induction on the complexity of S. For all statements S other than a procedure call P, this can be done using the well-known techniques (see [A] or [B]). To establish the case $S \equiv P$, however, the first step is to derive $\{\exists n\, p_0(n)\}P\{\overline{x} = \overline{z}\}$. Here the elimination rule is applied. From this latter correctness assertion arbitrary valid correctness formulas about the procedure P can be derived by an application of the invariance, substitution, and consequence rules. Having proved the derivability

of any valid correctness assertion $\{p\}S\{q\}$ from $\{p_0(n)\}P\{\overline{x} = \overline{z}\}$ it is shown by an application of the recursion rule that the latter assertion is derivable, thus establishing completeness.

Now note that this proof is not valid to establish the completeness of our new proof system T, because it uses both the elimination rule and the recursion with respect to the same variable n, which is not allowed, whether or not the variable n is a counter variable. Therefore we proceed in a different way. Instead of applying the elimination rule to derive $\{\exists n\, p_0(n)\}P\{\overline{x} = \overline{z}\}$, we use the recursion rule. Therefore we have to prove

$$\{p_0(n)\}P\{\overline{x} = \overline{z}\} \vdash_I \{p_0(n+1)\}S_0\{\overline{x} = \overline{z}\}. \tag{1}$$

A straightforward induction on the complexity of S_0 obviously does not work. To be able to carry out some inductive argument we prove the following generalised version:

$$\{p_0(n)\}P\{\overline{x} = \overline{z}\} \vdash_I \{Pre(S, q, n)\}S\{q\}.$$

for arbitrary S and q. This is done in lemma 5.5. Then, substituting S_0 for S and $\overline{x} = \overline{z}$ for q, and applying the consequence rule, it is possible to prove (1), so that the recursion rule can be applied.

The rest of this section provides the details.

Lemma 5.4
For any first-order formula q, for any variable n, and for any sequences $\overline{v}$ and $\overline{w}$ of distinct variables such that $\overline{v} \cap \overline{w} = \overline{v} \cap Var(S_0) = \overline{w} \cap Var(S_0) = \overline{v} \cap Count = \overline{w} \cap Count = \emptyset$, we have

$$\models_I Pre(P, q, n)[\overline{v}/\overline{w}] \rightarrow Pre(P, q[\overline{v}/\overline{w}], n).$$

Proof
From the definition of $Pre(P, q, n)$ it follows that

1. For every $k \in \mathbf{N}$ we have $\models_I \{Pre(P, q, n) \wedge n = \underline{k}\}P^{[k]}\{q\}$.
2. $\models_I \{p\}P^{[k]}\{q\}$ implies $\models_I p \wedge n = \underline{k} \rightarrow Pre(P, q, n)$.

From 1 it follows by the soundness of the substitution rule that for every $k \in \mathbf{N}$

$$\models_I \{(Pre(P, q, n) \wedge n = \underline{k})[\overline{v}/\overline{w}]\}P^{[k]}\{q[\overline{v}/\overline{w}]\}.$$

Note that $(Pre(P, q, n) \wedge n = \underline{k})[\overline{v}/\overline{w}] \equiv Pre(P, q, n)[\overline{v}/\overline{w}] \wedge n = \underline{k}$. So we have by 2 for arbitrary $k \in \mathbf{N}$

$$\models_I Pre(P, q, n)[\overline{v}/\overline{w}] \wedge n = \underline{k} \rightarrow Pre(P, q[\overline{v}/\overline{w}], n).$$

Therefore we conclude

$$\models_I Pre(P, q, n)[\overline{v}/\overline{w}] \rightarrow Pre(P, q[\overline{v}/\overline{w}], n).$$

□

Lemma 5.5
For any program S and any first-order formula q,

$$\{p_0(n)\}P\{\overline{x} = \overline{z}\} \vdash_I \{Pre(S, q, n)\}S\{q\}.$$

Proof
The proof proceeds by induction on the length of S. We distinguish several cases:

- $S \equiv x := t$
  Within the proof system we can reason as follows:

  1. $\{q[t/x]\}x := t\{q\}$, by the assignment axiom.
  2. $\{Pre(x := t, q, n)\}x := t\{q\}$ by the consequence rule.

  In order to justify step 2 above, we prove

  $$\models_I Pre(x := t, q, n) \rightarrow q[t/x].$$

  Let $\sigma \models_I Pre(x := t, q, n)$, then for $\sigma' = M_I(S^{[k]})(\sigma)$, where $k = \sigma(n)$, we have $\sigma' \models_I q$. But $S^{[k]} \equiv S$, so $\sigma' = \sigma\{\sigma(t)/x\}$, and therefore, by proposition 2.5, we get $\sigma \models_I q[t/x]$.

- $S \equiv P$
  Let $\overline{u}$ be a sequence (of the same length as $\overline{z}$) of fresh variables, not occurring in $Pre(P, q, n)$ or S_0, such that $\overline{u} \cap Count = \emptyset$, and take $q_1 \equiv q[\overline{u}/\overline{z}]$. Now we can reason within the proof system as follows:

  1. $\{p_0(n)\}P\{\overline{x} = \overline{z}\}$, the assumption.
  2. $\{p_0(n) \wedge q_1[\overline{z}/\overline{x}]\}P\{\overline{x} = \overline{z} \wedge q_1[\overline{z}/\overline{x}]\}$, by the invariance rule.
  3. $\{p_0(n) \wedge q_1[\overline{z}/\overline{x}]\}P\{q_1\}$, by the consequence rule (note that $\models_I \overline{x} = \overline{z} \wedge q_1[\overline{z}/\overline{x}] \rightarrow q_1$).
  4. $\{\exists\overline{z}(p_0(n) \wedge q_1[\overline{z}/\overline{x}])\}P\{q_1\}$, by the elimination rule.
  5. $\{Pre(P, q_1, n)\}P\{q_1\}$, by the consequence rule. This will be justified below.
  6. $\{Pre(P, q, n)[\overline{u}/\overline{z}]\}P\{q_1\}$, again by the consequence rule, now making use of lemma 5.4.
  7. $\{Pre(P, q, n)\}P\{q\}$, by the substitution rule.

  In order to justify step 5, we still have to show that

  $$\models_I Pre(P, q_1, n) \rightarrow \exists\overline{z}(p_0(n) \wedge q_1[\overline{z}/\overline{x}]).$$

  Let $\sigma \models_I Pre(P, q_1, n)$. Then for $\sigma' = M_I(P^{[k]})(\sigma)$, where $k = \sigma(n)$, we have $\sigma' \models_I q_1$.
  Let $\overline{d} = \sigma'(\overline{x})$. Then, since $\overline{z} \cap Var(S_0) = \emptyset$, we get $\sigma'\{\overline{d}/\overline{z}\} = M_I(P^{[k]})(\sigma\{\overline{d}/\overline{z}\})$. Furthermore $\sigma'\{\overline{d}/\overline{z}\} \models_I \overline{x} = \overline{z}$, so

  $$\sigma\{\overline{d}/\overline{z}\} \models_I p_0(n). \tag{2}$$

Now by proposition 2.5 we have

$$\sigma\{\overline{d}/\overline{z}\} \models_I q_1[\overline{z}/\overline{x}] \quad \text{iff} \quad \sigma\{\overline{d}/\overline{z}\}\{\overline{d}/\overline{x}\} \models_I q_1. \tag{3}$$

Note that $\sigma'(y) = \sigma\{\overline{d}/\overline{z}\}\{\overline{d}/\overline{x}\}(y)$ for all $y \in FV(q_1)$. So from $\sigma' \models_I q_1$ and from (3) we can infer by proposition 2.6 that

$$\sigma\{\overline{d}/\overline{z}\} \models_I q_1[\overline{z}/\overline{x}]. \tag{4}$$

From (2) and (4) we conclude

$$\sigma \models_I \exists \overline{z}(p_0(n) \wedge q_1[\overline{z}/\overline{x}]).$$

- $S \equiv S_1; S_2$
  Again we reason within the proof system:

  1. $\{p_0(n)\}P\{\overline{x} = \overline{z}\}$, the assumption.
  2. $\{Pre(S_2, q, n)\}S_2\{q\}$, from 1 by the induction hypothesis.
  3. $\{Pre(S_1, Pre(S_2, q, n), n)\}S_1\{Pre(S_2, q, n)\}$, from 1 by the induction hypothesis.
  4. $\{Pre(S_1, Pre(S_2, q, n), n)\}S_1; S_2\{q\}$, by the sequential composition rule from 2 and 3.
  5. $\{Pre(S_1; S_2, q, n)\}S_1; S_2\{q\}$, by the consequence rule. It is easy to see that

     $$\models_I Pre(S_1; S_2, q, n) \rightarrow Pre(S_1, Pre(S_2, q, n), n).$$

- $S \equiv$ **if** b **then** S_1 **else** S_2 **fi**
  We reason within the proof system:

  1. $\{p_0(n)\}P\{\overline{x} = \overline{z}\}$, the assumption.
  2. $\{Pre(S_1, q, n)\}S_1\{q\}$, from 1 by the induction hypothesis.
  3. $\{Pre(S, q, n) \wedge b\}S_1\{q\}$, by the consequence rule, using

     $$\models_I Pre(S, q, n) \wedge b \rightarrow Pre(S_1, q, n).$$

     (Note that $S^{[k]} =$ **if** b **then** $S_1^{[k]}$ **else** $S_2^{[k]}$ **fi**.)
  4. $\{Pre(S_2, q, n)\}S_2\{q\}$, from 1 by the induction hypothesis.
  5. $\{Pre(S, q, n) \wedge \neg b\}S_2\{q\}$, by the consequence rule, using

     $$\models_I Pre(S, q, n) \wedge \neg b \rightarrow Pre(S_2, q, n).$$

  6. $\{Pre(S, q, n)\}S\{q\}$, by the conditional rule from 3 and 5.

- $S \equiv$ **while** b **do** S_1 **od**

  We may assume that there exists a formula $\psi(m,n)$, where $m \notin \mathit{Count}$, such that $\sigma \models_I \psi(m,n)$ iff there exists a $\sigma' = M_I((S_1^{\{l\}})^{[k]})(\sigma) \neq \bot$ such that $\sigma' \models_I q$ and $M_I((S_1^{\{l'\}})^{[k]})(\sigma) = \bot$ for all $l' < l$, where $l = \sigma(m)$ and $k = \sigma(n)$. Here $S_1^{\{l\}}$ is defined with respect to the boolean expression b (see definition 2.1). The existence of such a formula $\psi(m,n)$ can be proved by the same techniques as used to prove lemma 5.2.

  Now we reason as follows within the proof system:

  1. $\{p_0(n)\}P\{\overline{x} = \overline{z}\}$, the assumption.
  2. $\{\mathit{Pre}(S_1, \psi(m,n), n)\}S_1\{\psi(m,n)\}$, from 1 by the induction hypothesis.
  3. $\{\psi(m+1,n)\}S_1\{\psi(m,n)\}$, by the consequence rule; we shall justify this below.
  4. $\{\exists m\, \psi(m,n)\}S\{\psi(0,n)\}$, by the iteration rule. Note that $\models_I \psi(0,n) \rightarrow \neg b$ and $\models_I \psi(m+1,n) \rightarrow b$. The truth of the first implication follows from the observation that for arbitrary $k \in \mathbf{N}$ we have $(S_1^{\{0\}})^{[k]} =$ **if** b **then** Ω **else skip fi**. The truth of the second one can be justified as follows: Let $\sigma \models_I \psi(m+1,n) \wedge \neg b$. From $\sigma \models_I \neg b$ we derive $\sigma = M_I((S_1^{\{0\}})^{[k]})(\sigma)$, where $\sigma(n) = k$. But by $\sigma \models_I \psi(m+1,n)$ we have $M_I((S_1^{\{l\}})^{[k]})(\sigma) = \bot$ for arbitrary $l < \sigma(m) + 1$. Contradiction.
  5. $\{\mathit{Pre}(S,q,n)\}S\{q\}$, by the consequence rule, making use of $\models_I \mathit{Pre}(S,q,n) \rightarrow \exists m\, \psi(m,n)$ and $\models_I \psi(0,n) \rightarrow q$. The truth of the first implication follows from the observation that for arbitrary $k \in \mathbf{N}$ if $\sigma' = M_I(S^{[k]})(\sigma)$ then there exists a $l \in \mathbf{N}$ such that $\sigma' = M_I((S_1^{\{l\}})^{[k]})(\sigma)$ and $M_I((S_1^{\{l'\}})^{[k]})(\sigma) = \bot$ for every $l' < l$. The truth of the second implication follows from the definition $S_1^{\{0\}} =$ **if** b **then** Ω **else skip fi**.

  We still have to justify step 3. To do so we prove

$$\models_I \psi(m+1,n) \rightarrow \mathit{Pre}(S_1, \psi(m,n), n).$$

  Let $\sigma \models_I \psi(m+1,n)$. Then for $\sigma' = M_I((S_1^{\{l+1\}})^{[k]})(\sigma)$ we have $\sigma' \models_I q$, where $l = \sigma(m)$ and $k = \sigma(n)$, while $M_I((S_1^{\{l'\}})^{[k]})(\sigma) = \bot$ for $l' < l+1$. Among other things, this implies that $\sigma \models_I b$.

  Now if we take $\sigma'' = M_I(S_1^{[k]})(\sigma)$, it follows that $\sigma' = M_I((S_1^{\{l\}})^{[k]})(\sigma'')$, and that $M_I((S_1^{\{l'\}})^{[k]})(\sigma'') = \bot$ for $l' < l$. Furthermore note that $l = \sigma(m) = \sigma''(m)$ and $k = \sigma(n) = \sigma''(n)$. So we have $\sigma'' \models_I \psi(m,n)$. Therefore

$$\sigma \models_I \mathit{Pre}(S_1, \psi(m,n), n).$$

This concludes the proof of lemma 5.5. □

Theorem 5.6
The proof system T is complete, that is, for any arithmetical interpretation I and any correctness formula $\{p\}S\{q\}$ we have

$$\models_I \{p\}S\{q\} \quad \text{implies} \quad \vdash_I \{p\}S\{q\}.$$

Proof
The proof proceeds by induction on the length of S. We present only the case $S \equiv P$. The other ones are treated exactly the same as in the proof of the completeness theorem for the sublanguage consisting of all those programs in which there occur no procedure calls, for which we can safely refer to [A], for example.

So let us assume $\models_I \{p\}P\{q\}$. We have to show that $\vdash_I \{p\}P\{q\}$. We know from lemma 5.5 that

$$\{p_0(n)\}P\{\overline{x} = \overline{z}\} \vdash_I \{Pre(S_0, \overline{x} = \overline{z}, n)\}S_0\{\overline{x} = \overline{z}\}.$$

From $P^{[k+1]} \equiv P[S_0^{(k+1)}/P] \equiv S_0^{(k+1)} \equiv S_0[S_0^{(k)}/P] \equiv S_0^{[k]}$ together with definition 5.3 and lemma 5.2 we derive

$$\models_I p_0(n+1) \to Pre(S_0, \overline{x} = \overline{z}, n).$$

This enables us to apply the consequence rule to derive

$$\{p_0(n)\}P\{\overline{x} = \overline{z}\} \vdash_I \{p_0(n+1)\}S_0\{\overline{x} = \overline{z}\}.$$

Furthermore $\models_I \neg p_0(0)$ because $P^{[0]} \equiv \Omega$. So applying the recursion rule yields

$$\vdash_I \{\exists n\, p_0(n)\}P\{\overline{x} = \overline{z}\}.$$

Now let $q_1 \equiv q[\overline{u}/\overline{z}]$, where $\overline{u}$ is a sequence of fresh, distinct variables of the same length as $\overline{z}$, such that $\overline{u} \cap Count = \emptyset$. We apply the invariance rule, yielding:

$$\vdash_I \{\exists n\, p_0(n) \wedge q_1[\overline{z}/\overline{x}]\}P\{\overline{x} = \overline{z} \wedge q_1[\overline{z}/\overline{x}]\}.$$

Note that $\models_I \overline{x} = \overline{z} \wedge q_1[\overline{z}/\overline{x}] \to q_1$, so applying the consequence rule gives us

$$\vdash_I \{\exists n\, p_0(n) \wedge q_1[\overline{z}/\overline{x}]\}P\{q_1\}.$$

Now note that $\overline{z} \cap Count = \emptyset$ and $\overline{z} \cap FV(q_1) = \emptyset$. Therefore application of the elimination rule yields

$$\vdash_I \{\exists \overline{z}(\exists n\, p_0(n) \wedge q_1[\overline{z}/\overline{x}])\}P\{q_1\}.$$

We shall show below that

$$\models_I p_1 \to \exists \overline{z}(\exists n\, p_0(n) \wedge q_1[\overline{z}/\overline{x}]) \tag{5}$$

where $p_1 \equiv p[\overline{u}/\overline{z}]$. Applying the consequence rule thus yields $\vdash_I \{p_1\}P\{q_1\}$. Finally we apply the substitution rule, yielding the desired result:

$$\vdash_I \{p\}P\{q\}.$$

We still have to prove (5). From the soundness of the substitution rule it follows that

$$\models_I \{p\}P\{q\} \quad \text{implies} \quad \models_I \{p_1\}P\{q_1\}$$

(this also follows easily from propositions 2.5 and 2.7).

Now let $\sigma \models_I p_1$, then it follows that $\sigma' \models_I q_1$ for $\sigma' = M_I(P)(\sigma)$. Since $\sigma' = \bigsqcup_k M_I(S_0^{(k)})(\sigma) = \bigsqcup_k M_I(P^{[k]})(\sigma) \neq \bot$ (see definition 2.1) there must be a $k \in \mathbf{N}$ such that $\sigma' = M_I(P^{[k]})(\sigma)$. Let furthermore $\overline{d} = \sigma'(\overline{x})$. Then, from the definition of $p_0(n)$, it follows that

$$\sigma\{\overline{d}/\overline{z}\}\{k/n\} \models_I p_0(n).$$

Therefore

$$\sigma\{\overline{d}/\overline{z}\} \models_I \exists n\, p_0(n). \tag{6}$$

From $\sigma' \models_I q_1$ and, (since $\overline{z} \cap FV(q_1) = \emptyset$, $q_1 = q_1[\overline{z}/\overline{x}][\overline{x}/\overline{z}]$) we deduce by proposition 2.5

$$\sigma'\{\overline{d}/\overline{z}\} \models_I q_1[\overline{z}/\overline{x}].$$

Now since $FV(q_1[\overline{z}/\overline{x}]) \cap Var(S_0) = \emptyset$ we get by proposition 2.7 that $\sigma\{\overline{d}/\overline{z}\}(y) = \sigma'\{\overline{d}/\overline{z}\}(y)$ for all $y \in (FV(q_1[\overline{z}/\overline{x}])$. So proposition 2.6 gives us

$$\sigma\{\overline{d}/\overline{z}\} \models_I q_1[\overline{z}/\overline{x}]. \tag{7}$$

Now from (6) and (7) we conclude

$$\sigma \models_I \exists \overline{z}(\exists n p_0(n) \wedge q_1[\overline{z}/\overline{x}]).$$

This concludes the proof of (5) and also of theorem 5.6. □

6 Application to dynamic logic

In this section we discuss the relevance of our analysis to dynamic logic [Ha]. Whereas in Hoare logic programs and logical formulas are strictly separated, in dynamic logic programs can occur *inside* logical formulas. There they play the role of *modalities*, i.e., they talk about the truth of a formula in other states than the current one. Formally we have the following definition of the assertion language of dynamic logic:

Definition 6.1
The set of dynamic logic assertions, with typical element p, is given by the following grammar:

$$\begin{array}{lcll} p & ::= & q & \text{where } q \text{ is an atomic assertion} \\ & | & p_1 \wedge p_2 \mid \ldots & \\ & | & \forall x\, p & \\ & | & \langle S\rangle p \mid [S]p & \text{where } S \text{ is a statement.} \end{array}$$

An atomic assertion consists of a predicate symbol of our first-order language L (see section 2.1) applied to a number of terms. We take the same syntax for statements as defined in section 2.1.

Definition 6.2
Let I be an interpretation. For an assertion p and a state $\sigma \in \Sigma(I)$ we define the truth of p in σ, denoted by $\sigma \models_I p$, as follows:

- If q is an atomic first-order assertion, $\sigma \models_I q$ is defined as usual (cf. section 2.2).
- $\sigma \models_I p_1 \wedge p_2$ iff $\sigma \models_I p_1$ and $\sigma \models_I p_2$. Analogously for the other propositional connectives.
- $\sigma \models_I \forall x\, p$ iff for every $d \in I_D$ we have $\sigma\{d/x\} \models_I p$.
- $\sigma \models_I \langle S \rangle p$ iff there exists a $\sigma' \neq \perp$ such that $\sigma' = M_I(S)(\sigma)$ and $\sigma' \models_I p$,
- $\sigma \models_I [S]p$ iff for all $\sigma' \neq \perp$ such that $\sigma' = M_I(S)(\sigma)$ we have $\sigma' \models_I p$.

So the partial correctness interpretation of the Hoare triple $\{p\}S\{q\}$ can be rendered in dynamic logic by the assertion $p \rightarrow [S]q$, and its total correctness interpretation corresponds to the assertion $p \rightarrow \langle S \rangle q$. Now in [Ha] a proof system based on dynamic logic is presented which is sound and complete (for *arithmetical* interpretations). In this proof system the problem we analyzed with respect to the Hoare-style proof system is solved in a different way. The main idea consists of extending the programming language by interpreting first-order assertions as programs in the following manner:

Definition 6.3
Let p be a first-order assertion with free variables $\overline{x}$ and $\overline{y}$. Then we extend the syntax of statements (see section 2.1) by adding the clause $S ::= p^{\overline{x}}_{\overline{y}}$. Now if I is an interpretation we define the meaning of the program $p^{\overline{x}}_{\overline{y}}$ as follows:

$$M_I(p^{\overline{x}}_{\overline{y}})(\sigma) = \{\, \sigma' : \exists \overline{d}(\sigma' = \sigma\{\overline{d}/\overline{x}\} \wedge \sigma\{\overline{d}/\overline{y}\} \models_I p \,\}.$$

Note that we thus have introduced nondeterminism in our programming language: A single statement can be executed in several ways, possibly leading to different results. It is straightforward to modify the meaning function M_I as given in definition 2.1 in order to cope with nondeterminism.

Example
Let $p \equiv y = x + 1$. Furthermore, let I be an interpretation such that I_D is the set of integers together with the standard interpretation of the arithmetical operations. We then have

$$\begin{aligned} M_I(p^x_y)(\sigma) &= \{\, \sigma' : \exists n(\sigma' = \sigma\{n/x\} \wedge \sigma\{n/y\} \models_I y = x + 1\} \\ &= M_I(x := x + 1)(\sigma). \end{aligned}$$

In general we have that $M_I(p^{\overline{x}}_{\overline{y}}) = M_I(S)$ formalises that $p(\overline{x}, \overline{y})$ describes the graph of S as explained in the previous section, assuming that S is a program with $Var(S, S_0) = \overline{x}$.

We now give a proof system based on dynamic logic for our programming language along the lines of [Ha]. In order to focus on recursion we omit the iterative command.

Definition 6.4
The proof system H consists of the following axioms:

Assignment 1: $[x := t]p \leftrightarrow p[t/x]$

where p is a first-order assertion.

Assignment 2: $\langle x := t\rangle p \leftrightarrow p[t/x]$

where p is a first-order assertion.

Sequential composition 1: $[S_1; S_2]p \leftrightarrow [S_1][S_2]p$

Sequential composition 2: $\langle S_1; S_2\rangle p \leftrightarrow \langle S_1\rangle\langle S_2\rangle p$

Conditional 1: $[\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}]p \leftrightarrow (b \rightarrow [S_1]p \wedge \neg b \rightarrow [S_2]p)$

Conditional 2: $\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}\rangle p \leftrightarrow (b \rightarrow \langle S_1\rangle p \wedge \neg b \rightarrow \langle S_2\rangle p)$

Assertion 1: $[p^{\overline{z}}_{\overline{y}}]q \leftrightarrow \forall \overline{z}(p[\overline{z}/\overline{y}] \rightarrow q[\overline{z}/\overline{x}])$

Assertion 2: $\langle p^{\overline{z}}_{\overline{y}}\rangle q \leftrightarrow \exists \overline{z}(p[\overline{z}/\overline{y}] \wedge q[\overline{z}/\overline{x}])$

Invariance 1: $p \rightarrow [P]p$

where $Var(p) \cap Var(S_0) = \emptyset$.

Invariance 2: $(p \rightarrow \langle P\rangle q) \rightarrow (p \wedge r \rightarrow \langle P\rangle p \wedge r)$

where $Var(r) \cap Var(S_0) = \emptyset$.

$\forall$-Elimination $\forall x\, p \rightarrow p$

Further, we have the following rules:

Modus ponens: $\dfrac{p \rightarrow q,\ q}{q}$

Box:
$$\frac{p \to q}{[S]p \to [S]q}$$

Diamond:
$$\frac{p \to q}{\langle S\rangle p \to \langle S\rangle q}$$

Universal:
$$\frac{p \to q}{\forall x\, p \to \forall x\, q}$$

$\forall$-Introduction:
$$\frac{p}{\forall x\, p}$$

Recurse 1:
$$\frac{\overline{x} = \overline{y} \to [S_0[p^{\overline{x}}_{\overline{y}}/P]](p[\overline{y}/\overline{x}, \overline{x}/\overline{y}])}{\overline{x} = \overline{y} \to [P](p[\overline{y}/\overline{x}, \overline{x}/\overline{y}])}$$

where $Var(S_0) = \overline{x}$.

Recurse 2:
$$\frac{p(n+1) \to \langle S_0[p(n)^{\overline{x}}_{\overline{y}}/P]\rangle \overline{x} = \overline{y} \qquad \neg p(0)}{\exists n\, p(n) \to \langle P\rangle \overline{x} = \overline{y}}$$

where $Var(S_0) = \overline{x}$ and $n \notin Var(S_0) \cup \overline{x} \cup \overline{y}$.

Given an interpretation I we denote by $\vdash_I$ provability in the proof system that can be obtained by adding to the above axioms and rules all the first-order assertions that valid with respect to I.

Especially the recursion rules need some explanation. The key to understanding these rules is the following theorem:

Theorem 6.5
Let I be some interpretation and let S be a statement such that $Var(S, S_0) = \overline{x}$ and $\overline{x} \cap \overline{y} = \emptyset$. Then we have the following two equivalences:

$$\models_I \overline{x} = \overline{y} \to [S](p[\overline{y}, \overline{x}/\overline{x}, \overline{y}]) \qquad \text{iff} \qquad M_I(S) \subseteq M_I(p^{\overline{x}}_{\overline{y}}) \tag{8}$$

$$\models_I p \to \langle S\rangle \overline{x} = \overline{y} \qquad \text{iff} \qquad M_I(p^{\overline{x}}_{\overline{y}}) \subseteq M_I(S) \tag{9}$$

Proof
See [Ha]. □

The first equivalence states that the assertion $\overline{x} = \overline{y} \to [S](p[\overline{y}, \overline{x}/\overline{x}, \overline{y}])$ holds if and only if the graph of the statement S is contained in that of $p^{\overline{x}}_{\overline{y}}$. So to prove the assertion $\overline{x} = \overline{y} \to [P](p[\overline{y}, \overline{x}/\overline{x}, \overline{y}])$ amounts to showing that $M_I(P) \subseteq M_I(p^{\overline{x}}_{\overline{y}})$, or, equivalently, that for all k we have $M_I(S_0^{(k)}) \subseteq M_I(p^{\overline{x}}_{\overline{y}})$. To prove this by induction we have to show that $M_I(S_0^{(k+1)}) \subseteq M_I(p^{\overline{x}}_{\overline{y}})$. But by the induction hypothesis we have $M_I(S_0^{(k)}) \subseteq M_I(p^{\overline{x}}_{\overline{y}})$, from which it follows that $M_I(S_0^{(k+1)}) \subseteq M_I(S_0[p^{\overline{x}}_{\overline{y}}/P])$ (cf. proposition 2.3). So it suffices to show that $M_I(S_0[p^{\overline{x}}_{\overline{y}}/P]) \subseteq M_I(p^{\overline{x}}_{\overline{y}})$. Using the equivalence (8) this amounts to proving the assertion $\overline{x} = \overline{y} \to [S_0[p^{\overline{x}}_{\overline{y}}/P]]p[\overline{y}, \overline{x}/\overline{x}, \overline{y}]$.

In a similar way we can give a justification of the rule Recurse 2, using the equivalence (9).

We have the following theorems about this proof system:

Theorem 6.6 (Soundness)
Let I be an arithmetical interpretation. For an arbitrary dynamic logic assertion p we have

$$\vdash_I p \qquad \text{implies} \qquad \models_I p.$$

Proof
See [Ha]. □

Theorem 6.7 (Completeness)
Let I be an arithmetical interpretation. For an arbitrary dynamic logic assertion p we have

$$\models_I p \qquad \text{implies} \qquad \vdash_I p.$$

Proof
See [Ha]. □

Note that this proof system does not have rules corresponding to the substitution rule and the elimination rule of our Hoare-style proof system. These rules are in fact in a way incorporated by the rules Assertion 1 and Assertion 2. However the resulting proof system is quite complicated. We will show that there also exists a sound and complete proof system based on dynamic logic which more closely corresponds to the Hoare-style proof systems, and that we do not need to extend our programming language by interpreting assertions as programs. We will make use of the fact that we can express the special role of the counter variables in the Hoare style proof system directly in dynamic logic. We first introduce the following new version of Recurse 1:

Definition 6.8
We have the following rule dealing with partial correctness of recursion:

Recursion 1: $$\frac{\forall \overline{z}(p \to [P]q) \to \forall \overline{z}(p \to [S_0]q)}{p \to [P]q}$$

where $\overline{z} = Var(S_0, p, q)$.

Next we present the following version of Recurse 2:

Definition 6.9
We have the following rule dealing with total correctness of recursion:

Recursion 2: $$\frac{\forall \overline{z}(p(n) \to \langle P \rangle q) \to \forall \overline{z}(p(n+1) \to \langle S_0 \rangle q) \qquad \neg p(0)}{\exists n\, p(n) \to \langle P \rangle q}$$

where $Var(S_0, p(n), q) \setminus \{n\} \subseteq \overline{z}$ and $n \notin \overline{z}$.

Note that universally quantifying all the variables except the variable n corresponds to the different interpretation of the counter variables. Furthermore we have the following versions of the elimination and the substitution rules:

Definition 6.10
We have the following two elimination axioms:

Elimination 1: $\forall z(p \to [P] \to (\exists z p \to [P]q)$

and

Elimination 2: $\forall z(p \to \langle P\rangle q) \to (\exists z p \to \langle P\rangle q)$

where in both rules we require that $z \notin Var(S_0, q)$.

Definition 6.11
We have the following two substitution axioms:

Substitution 1: $\forall z(p \to \langle P\rangle q) \to \forall y(p[y/z] \to \langle P\rangle q[y/z])$

and

Substitution 2: $\forall z(p \to [P]q) \to \forall y(p[y/z] \to [P]q[y/z])$

where in both rules we require that $z, y \notin Var(S_0)$ and $y \notin Var(p, q)$.

This new system can be proved to be sound by a straightforward induction on the length of the derivation. The soundness of the rule Recursion 1 is established in a similar way as the corresponding rule of the system manipulating correctness phrases introduced to prove the soundness of the Hoare-style proof system for partial correctness (see [A]). The soundness of the rule Recursion 2 is established in a similar way as the corresponding rule of the system manipulating correctness phrases as defined in definition 3.3.

Completeness follows from the following theorem, see [Ha]:

Theorem 6.12
Let I be an arithmetical interpretation and let p and q be first-order assertions. We have

$$\models_I p \to \langle S\rangle q \quad \text{implies} \quad \vdash_I p \to \langle S\rangle q$$

and

$$\models_I p \to [S]q \quad \text{implies} \quad \vdash_I p \to [S]q.$$

The proofs of both implications follow the proof method for the completeness of the corresponding Hoare-style proof systems, i.e., the Hoare-style proof system for partial correctness and the system presented in the previous section. We illustrate this by the proof of the lemma corresponding to lemma 5.5. Let $Pre(S, q, n)$ and $p_0(n)$ be defined as in the previous section.

Lemma 6.13
Let I be an arithmetical interpretation. For every statement S and first-order assertion q we have

$$\vdash_I \forall \overline{y}(p_0(n) \to \langle P\rangle \overline{x} = \overline{z}) \to \forall \overline{y}(Pre(S, q, n) \to \langle S\rangle q)$$

where $Var(p_0(n), q, S) \cup \overline{x} \cup \overline{z} \setminus \{n\} \subseteq \overline{y}$, $n \notin \overline{y}$, and $\overline{x} = Var(S_0)$.

Proof

The proof proceeds by induction on the complexity of S. Here we only deal with the case of $S \equiv P$; let $q_1 \equiv q[\overline{u}/\overline{z}]$, where $\overline{u}$ are some new variables:

1. $(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow (p_0(n) \wedge q_1[\overline{z}/\overline{x}] \rightarrow \langle P \rangle \overline{x} = \overline{z} \wedge q_1[\overline{z}/\overline{x}])$, by Invariance 2.
2. $(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow (p_0(n) \wedge q_1[\overline{z}/\overline{x}] \rightarrow \langle P \rangle q_1)$, from $\models_I \overline{x} = \overline{z} \wedge q_1[\overline{z}/\overline{x}] \rightarrow q_1$, using Diamond and some propositional reasoning.
3. $\forall \overline{z}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow \forall \overline{z}(p_0(n) \wedge q_1[\overline{z}/\overline{x}] \rightarrow \langle P \rangle q_1)$, by Universal.
4. $\forall \overline{z}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow (\exists \overline{z}(p_0(n) \wedge q_1[\overline{z}/\overline{x}]) \rightarrow \langle P \rangle q_1)$, by Elimination 2 and some propositional reasoning.
5. $\forall \overline{z}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow (Pre(P, q_1, n) \rightarrow \langle P \rangle q_1)$, because $\models_I Pre(P, q_1, n) \rightarrow \exists \overline{z}(p_0(n) \wedge q_1[\overline{z}/\overline{x}])$ (see the previous section) and using some propositional reasoning.
6. $\forall \overline{z}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow (Pre(P, q, n)[\overline{u}/\overline{z}] \rightarrow \langle P \rangle q_1)$, since $\models_I Pre(P, q, n)[\overline{u}/\overline{z}] \rightarrow Pre(P, q_1, n)$ (see lemma 5.4) and using some propositional reasoning.
7. $\forall \overline{z}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow \forall \overline{u}(Pre(P, q, n)[\overline{u}/\overline{z}] \rightarrow \langle P \rangle q_1)$, by Universal, $\forall$-Elimination (note that $\overline{u} \cap Var(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) = \emptyset$), and using some propositional reasoning.
8. $\forall \overline{z}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z})$ $rightarrow \forall \overline{z}(Pre(P, q, n) \rightarrow \langle P \rangle q)$, by Substitution 2 and some propositional reasoning.
9. $\forall \overline{y}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow \forall \overline{y}(Pre(P, q, n) \rightarrow \langle P \rangle q)$, by Universal.

□

From this lemma we derive in the same way as in the previous section that

$$\forall \overline{y}(p_0(n) \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow \forall \overline{y}(p_0(n+1) \rightarrow \langle S_0 \rangle \overline{x} = \overline{z}).$$

Applying the recursion rule then gives us the derivability of

$$\exists n\, p_0 \rightarrow \langle P \rangle \overline{x} = \overline{z}.$$

Next we apply the rule $\forall$-Introduction, which gives us the derivability of

$$\forall \overline{z}(\exists n\, p_0 \rightarrow \langle P \rangle \overline{x} = \overline{z}).$$

We have

$$\vdash_I \forall \overline{z}(\exists n\, p_0 \rightarrow \langle P \rangle \overline{x} = \overline{z}) \rightarrow \forall \overline{z}(p \rightarrow \langle S \rangle q)$$

for any valid assertion $p \rightarrow \langle S \rangle q$. The proof of this claim proceeds in a similar way as the one of lemma 6.13. So we have the derivability of

$$\forall \overline{z}(p \rightarrow \langle S \rangle q).$$

Using the $\forall$-Elimination axiom and the Modus ponens rule then concludes the completeness proof.

7 Conclusion

We have studied in this paper a well-known Hoare-style proof system for the total correctness of recursive procedures. We showed that the proof system as presented in the literature is unsound due to the incorrect interaction of the recursion rule and the rules which formalise reasoning about invariance properties. Our solution to this problem consisted in defining some appropriate restrictions on the applicability of those rules which can interact in an incorrect way. We proved the system to be sound along the lines of [A] using a transformation of the system into a Gentzen-like calculus, thus turning it into a system in which the recursion rule is no longer a metarule. However, the interpretation of the result of this transformation differs from the one used in [A] to prove the soundness of the system for partial correctness. Special care had to be taken concerning the interpretation of the variables used to establish termination of procedures.

Furthermore we proved that even with these restrictions the proof system is still complete. The completeness proof differs quite substantially from the one given by [A] because there the restrictions on the applicability of some rules are not satisfied.

In [S] Sokołowski presented a different formulation of the recursion rule, based on *predicate transformers*, in order to solve the problem of how to interpret the notion of derivability in the premise of the recursion rule. In the conclusion of this new version the existential quantification of the variable used to establish termination is replaced by an *infinite* disjunction. As a consequence, our counterexample to the soundness of the system does not apply to this new version of the recursion rule. However the proof system based on predicate transformers transcends the framework of Hoare-style proof systems in allowing infinite disjunctions. Furthermore, the system presented in [S] is incomplete because it does not include a reasoning mechanism about invariance properties. A similar proof as given in [A] that the recursion rule for the partial correctness of recursive procedures does not suffice shows the same for the recursion rule for total correctness.

We also applied our analysis to dynamic logic. We constructed a proof system based on dynamic logic which more closely corresponds to the Hoare-style proof system than the one presented in [Ha]. In [Ha] the problems we encountered are solved in a way which complicates the proof system considerably. In particular it extends the programming language with assertions considered as statements. We showed how our technique to arrive at a sound and complete system based on Hoare logic can be formulated in the formalism of dynamic logic. The resulting proof rules are simpler and the programming language need not be extended.

References

[A] K.R. Apt: *Ten years of Hoare logic: a survey — part 1.* ACM Transactions on Programming Languages and Systems, Vol. 3, No. 4, 1981, pp. 431–483.

[B] J.W. de Bakker: *Mathematical theory of program correctness.* Prentice-Hall

International, 1980.

[Ha] D. Harel: *First-order dynamic logic*. Springer, Lecture Notes in Computer Science, Vol. 68, 1979.

[Ho] C.A.R. Hoare: *An axiomatic basis for computer programming*. Communications of the ACM, Vol. 12, No. 10, 1969, pp. 567–580,583.

[PM] A. Pnueli, Z. Manna (1982). *Verification of concurrent programs: the temporal framework*. In: R.S. Boyer, J.S. Moore (eds.): The Correctness Problem in Computer Science, Academic Press, 1982, pp. 215–273.

[S] S. Sokołowski: *Total correctness of procedures*. Proc. 6th Symp. Mathematical Foundations of Computer Science, Springer, Lecture Notes in Computer Science, Vol. 53, 1977, pp. 475–483.

[TZ] John V. Tucker, Jeffery I. Zucker: *Program correctness over abstract data types, with error-state semantics*. CWI Monographs, Vol. 6, North-Holland, 1988.