A New Perspective for
Hoare’s Logic and Peano’s Arithmetic ★

Zhaowei Xu

State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing, China, 100190
University of Chinese Academy of Sciences

Abstract

Hoare’s logic is an axiomatic system of proving programs correct, which has been extended to be a separation logic to reason about mutable heap structure. We develop the most fundamental logical structure of strongest postcondition of Hoare’s logic in Peano’s arithmetic \( PA \). Let \( p \in L \) and \( S \) be any while-program. The arithmetical definability of \( \mathbb{N} \)-computable function \( \mathcal{f}^N \) leads to separate \( S \) from \( SP(p, S) \), which defines the strongest postcondition of \( p \) and \( S \) over \( \mathbb{N} \), achieving an equivalent but more meaningful form in \( PA \). From the reduction of Hoare’s logic to \( PA \), together with well-defined underlying semantics, it follows that Hoare’s logic is sound and complete relative to the theory of \( PA \), which is different from the relative completeness in the sense of Cook. Finally, we discuss two ways to extend computability from the standard structure to nonstandard models of \( PA \).

Keywords: Hoare’s logic, Peano’s arithmetic, strongest postcondition, \( \mathbb{N} \)-computable function, definability, completeness

1. Introduction

Hoare’s logic is a formal system for the derivation of statements about the partial correctness of programs \([1]\); it was introduced by Hoare \([2]\) and

★Supported by the National Natural Science Foundation of China under Grant No. 61272135, and National Science and Technology Major Project of China under Grant No. 2012ZX01039-004.

Email address: xuzw@ios.ac.cn (Zhaowei Xu)
studied by Cook, Lipton, Wand and Clarke. Separation logic is a spatial logic for reasoning about mutable heap structure, which is an extension to Hoare’s logic to describe the applications of programs on the heap structure and the reasoning about memory update. It would be interesting to reconsider Hoare’s logic and the computability induced by the accessibility relations on models of Hoare’s logic, since Hoare’s logic, taken as a modal logic, defines its model on the assignments of the model $M'$ on the assignments of the model $M$ of the first-order languages based on which Hoare’s logic is defined, which leads to the special form of the completeness theorem different from that of the first-order logic.

Hoare’s logic for the set $WP$ of all while-programs with first-order assertion language $L$ and first-order specification $T$ we denote $HL(T)$. The formulas in Hoare’s system are triples of the form $\{p\}S\{q\}$, where $p, q \in L$ and $S \in WP$. We also call the triple $\{p\}S\{q\}$ a specified or asserted program. The specified program $\{p\}S\{q\}$ is true in a first-order model $M$, denoted by $M \models \{p\}S\{q\}$, if and only if for any initial assignment of the program variables over $M$, if $p$ holds and $S$ is executed, then either $S$ will fail to terminate or $q$ will be satisfied by the final assignment defined by the program modality over $M$. Call the language $L$ expressive relative to model $M$ if for all $p \in L$ and $S \in WP$ there exists $q \in L$ which defines the strongest postcondition of $p$ and $S$ relative to $M$. If $L$ is expressive relative to $M$, then $M \models \{p\}S\{q\}$ implies $HL(Th(M)) \vdash \{p\}S\{q\}$, where $p, q \in L$, $S \in WP$ and by $Th(M)$ we mean the set of all true $L$-sentences in $M$. This is the so-called relative completeness (i.e. relative to some model) in the sense of Cook.

In the following, let $L$ be the language of arithmetic. Since $L$ is expressive relative to the standard structure $N$, then any true specified program can be proved in $HL(Th(N))$. Bergstra and Tucker studied verification on an entirely proof-theoretic basis, where the facts about arithmetic must be formally deduced from Peano’s arithmetic $PA$ or its refinement, and not popped from the oracle $Th(N)$. The main theorem proved by them is listed as follows.

**Theorem 1.1** (Bergstra, et.al [1983]). Given an assertion $p \in L$ and program $S \in WP$ one can effectively calculate an assertion $SP(p, S) \in L$ such that

1. $SP(p, S)$ defines the strongest postcondition of $S$ relative to $p$ on the set of states over $N$;
2. $HL(PA) \vdash \{p\}S\{SP(p, S)\}$. And, for any refinement $T$ of Peano arithmetic, including $PA$ itself,
The requirement of expressiveness restricts the interpretation of \( PA \) to only the standard structure \( \mathbb{N} \) up to isomorphism [4]. Therefore, relative to \( \mathbb{N} \), Bergstra and Tucker proved that \( HL(PA) \) can be reduced to \( PA \), which is their original intent. Closer scrutiny of their argumentation reveals that Theorem 1.1 also holds relative to the theory of \( PA \) (i.e. relative to all models of \( PA \)) under the assumption that they generalized the classical coding function, which is used to code a finite sequence of natural numbers, to code an \( M \)-finite sequence of numbers for any model \( M \) of \( PA \) (i.e. generalized coding function), where by \( M \)-finite we mean the sequence has 'length' corresponding to elements in \( M \). In the sequel, we develop our theoretical results relative to the theory of \( PA \) under this assumption.

The semantical definition of strongest postcondition is well formalized in Cook[3, p85]. Bergstra, et.al[11, p271] gives the formal definition of strongest postcondition of \( p \in L \) and \( S \in WP \) relative to \( \mathbb{N} \), i.e. \( SP(p, S) \) mentioned above, and proves some of its important properties as a theorem in \( PA \). In this paper, we develop the most fundamental logical structure of strongest postcondition of Hoare’s logic applied to arithmetic as it is defined by Peano’s axioms, through which we can explain strongest postcondition more intuitively at the syntactic level. In principle, the \( \mathbb{N} \)-computability induced by while-programs is equivalent to that of recursive functions, which are arithmetically definable [12, p199], hence, we can arithmetically define \( \mathbb{N} \)-computable function \( f^N_S \) induced by while-program \( S \) concretely. The arithmetical definability of \( f^N_S \) leads to separate \( S \) from \( SP(p, S) \), achieving an equivalent but more meaningful form in \( PA \), which is illustrated as follows.

**Theorem 1.2** (Separation Theorem). For any \( n \)-place formula \( p(x) \in L \), and \( n \)-place program \( S \in WP \),

\[
PA \vdash SP(p, S)(\overrightarrow{x}) \leftrightarrow \exists \overrightarrow{u} \left( p(\overrightarrow{u} / \overrightarrow{x}) \land \alpha_S(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right),
\]

where \( \alpha_S(\overrightarrow{x}, \overrightarrow{y}) \) (resp. \( SP(p, S)(\overrightarrow{x}) \)) arithmetically defines \( f^N_S \) (resp. \( sp_N(p, S) \)), \( p(\overrightarrow{u} / \overrightarrow{x}) \), i.e. \( p(u_1/x_1, \ldots, u_n/x_n) \), stands for the result of substituting \( u_i \) for each free occurrence of \( x_i \) in \( p \) for all \( 1 \leq i \leq n \) and \( \alpha_S(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \) is defined analogously.

The Separation Theorem captures the essence of the expressiveness condition. Replacing \( SP(p, S) \) by \( \exists \overrightarrow{u} \left( p(\overrightarrow{u} / \overrightarrow{x}) \land \alpha_S(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right) \) in statement
one can not just obtain a conceptual clarity, but also simplify the strongest postcondition calculus inside considerably. Then it follows the following corollary.

**Corollary 1.3.** For any \( n \)-place formulas \( p(\vec{x}) \), \( q(\vec{x}) \in L \) and \( n \)-place program \( S \in WP \),

\[
HL(PA) \vdash \{ p \} S \{ q \} \iff PA \vdash p(\vec{x}) \land \alpha_S(\vec{x}, \vec{y}) \rightarrow q(\vec{y} / \vec{x}),
\]

where \( \alpha_S(\vec{x}, \vec{y}) \) arithmetically defines \( f^N_S \), \( q(\vec{y} / \vec{x}) \), i.e. \( q(y_1/x_1, \ldots, y_n/x_n) \), stands for the result of substituting \( y_i \) for each free occurrence of \( x_i \) in \( q \) for all \( 1 \leq i \leq n \).

The corollary reduces Hoare’s logic grounded on the theory of \( PA \) to the first-order logic \( PA \). If we can extend computability induced by \( S \) from the standard structure to nonstandard models of \( PA \) with the same \( \Sigma_1 \)-definability, that is \( M \)-computable function \( f^M_S \) is defined by \( \alpha_S(\vec{x}, \vec{y}) \) in nonstandard model \( M \) of \( PA \), then we can well define \( M \models \{ p \} S \{ q \} \) for any model \( M \models PA \), which we denote by \( HL(PA) \models \{ p \} S \{ q \} \). With the well-defined underlying semantics, it easily deduces that \( HL(PA) \models \{ p \} S \{ q \} \iff \) \( PA \models p(\vec{x}) \land \alpha_S(\vec{x}, \vec{y}) \rightarrow q(\vec{y} / \vec{x}) \). From Gödel’s Soundness and Completeness Theorem, together with Corollary 1.3, it follows that \( HL(PA) \models \{ p \} S \{ q \} \iff HL(PA) \vdash \{ p \} S \{ q \} \). Consequently, Hoare’s logic is sound and complete relative to the theory of \( PA \), which is different from the relative completeness. More remarks about how to build the underlying semantics are left for the last section.

The paper is organized as follows: In Preliminaries, we give the formal description of Peano’s arithmetic and coding functions. In section 3, we define Hoare’s logic based on \( PA \), its syntax and semantics. In section 4, we first give the definition of the \( N \)-computable function and argue its arithmetical definability; Then, we inductively construct \( SP(p, S) \) like that in \([11, p273]\); Finally, with the generalized coding function, we prove the Separation Theorem. In the last section, we discuss the extended computability and our future work.

2. Preliminaries

2.1. Peano’s arithmetic

By \( PA \) we denote the theory of Peano’s arithmetic, which is the set of all sentences of the language of arithmetic that are provable from Peano’s
axioms. Also, in this paper, we treat $PA$ as the logical system of Peano’s arithmetic. The desired data type semantics is the standard model $\mathbb{N}$. The domain of $\mathbb{N}$ is the set $\omega$ of natural numbers and its primitive operations are the addition $x + y$ and multiplication $x \cdot y$; $<$ is the linear ordering relation on $\mathbb{N}$; $0, 1 \in \omega$ are two distinguished elements. We shall use these notations for the function symbols, relation symbol and constant symbols in $L$. Let $L = \{+, \cdot, <, 0, 1\}$. List the axioms of Peano’s arithmetic as follows.

(S1) $x + 1 \neq 0$;
(S2) $x + 1 = y + 1 \Rightarrow x = y$;
(S3) $x + 0 = x$;
(S4) $x + (y + 1) = (x + y) + 1$;
(S5) $x \cdot 0 = 0$;
(S6) $x \cdot (y + 1) = (x \cdot y) + x$;
(S7) $\varphi(0, \vec{y}) \land \forall x (\varphi(x, \vec{y}) \rightarrow \varphi(x + 1, \vec{y})) \rightarrow \forall x \varphi(x, \vec{y})$, where $\varphi(x, \vec{y})$ is a formula in $L$.

For simplicity, we abbreviate $\underbrace{1 + \ldots + 1}_n$ as $n$. From the above Peano’s axioms, we may observe that equations (S3)-(S6) alone define $\mathbb{N}$ under initial algebra semantics and so we may consider (S1) and (S2) as additions, making a first refinement of the standard algebraic specification for arithmetic, desired to rule out finite models. The theoretical objective of adding the induction axiom (S7) is self-evident: one wants to generate all assertions which make statements about $\mathbb{N}$ which can be based on its simple arithmetical operators and which can be proved by the principle of induction. For example, one can obtain facts about the ordering $x < y$ of natural numbers by using the formula $\exists z (z \neq 0 \land x + z = y)$.

Let $M = (M, +^M, \cdot^M, <^M, 0^M, 1^M)$ be a countable model of Peano arithmetic, i.e. $M \models \varphi$ for each axiom $\varphi$ of $PA$, where $M$ is the universe of
M, and I is an interpretation such that

\[
\begin{align*}
I(+, M) &= +^M; \\
I(\cdot, M) &= \cdot^M; \\
I(<, M) &= <^M; \\
I(0, M) &= 0^M; \\
I(1, M) &= 1^M.
\end{align*}
\]

If M is not isomorphic to the standard model N = (N, +^N, \cdot^N, <^N, 0, 1), then M is nonstandard. There is a submodel N^M = (N^M, +^N, \cdot^N, <^N, 0^N, 1^N) of M such that N^M is isomorphic to N, where N^M = \{I(n, M) : n \in N\} is the universe of N^M and

\[
\begin{align*}
+^N &= +^M \upharpoonright N^M; \\
\cdot^N &= \cdot^M \upharpoonright N^M; \\
<^N &= <^M \upharpoonright N^M; \\
0^N &= 0^M; \\
1^N &= 1^M.
\end{align*}
\]

We call elements in N^M standard, and nonstandard otherwise. There exists a nonstandard model of PA, which is a direct consequence of the compactness theorem (see the argument of Corollary 25.3 in [12, p306]). For more information about nonstandard models, refer to Chapter 25 in [12].

In this paper, we need to understand the order in nonstandard models of PA. Let K be the set consisting of all natural numbers together with all pairs (q, a) where q is a rational number and a an integer. Let <_K be the order on K in which the natural numbers come first, in their usual order, and the pairs afterward, ordered as follows: (q, a) <_K (r, b) if and only if q < r in the usual order on rational numbers, or q = r and a < b in the usual order on integers. Then, the argument of Theorem 25.1 in [12, p304] implies that

Lemma 2.1.1. The order relation on any enumerable nonstandard model of PA is isomorphic to the ordering <_K of K.

The semantics of the first-order logic has its standard definition in model theory and we assume to be understood. The validity of p \in L over model M we write M \models p. Let T be a theory of L. We write T \models p to mean that for every M \models T, M \models p. From Gödel’s Soundness and Completeness Theorem, it deduces that T \models p if and only if T \models p.
2.2. Coding functions

A function \( f : \mathbb{N}^n \to \mathbb{N} \) is arithmetically definable, or arithmetical for short, if and only if there is a formula \( F(x,y) \) in \( L \) such that for any \( \bar{a} \in \mathbb{N}^n, b \in \mathbb{N} \) we have \( f(\bar{a}) = b \) if and only if \( \mathbb{N} \models F(\bar{a}, b) \). Boolos, et.al[12, p204] writes that:

**Lemma 2.2.1 (Definability of Recursive Functions).** *Every recursive function is arithmetical.*

It is a well-known fact that there exist one-to-one correspondences between the set of natural numbers and the set of ordered pairs of natural numbers and, indeed, that such a correspondence can be set up in an effective manner [13, p43].

**Lemma 2.2.2 (Pairing Function Lemma).** *There exist recursive functions \( \langle x, y \rangle, L(z), R(z) \) such that*

\[
\langle L(z), R(z) \rangle = z, \\
L(\langle x, y \rangle) = x, \\
R(\langle x, y \rangle) = y.
\]

For notational convenience, we denote \( (L(z), R(z)) \) by \( \bar{z} \). The pairing function can be extended to \( n \)-tuples by setting

\[
\langle x_1, x_2, \ldots, x_n \rangle = \langle x_1, \langle x_2, \ldots, x_n \rangle \rangle
\]

and

\[
\overline{x_1, \ldots, x_n} = (x_1, \overline{x_2, \ldots, x_n}).
\]

To code a finite sequence \( (a_0, \ldots, a_r) \) of natural numbers, we can use two suitably chosen natural numbers \( s \) and \( t \) as codes, which is the following lemma[14, p177].

**Lemma 2.2.3 (\( \beta \)-Function Lemma).** *There is a recursive function \( \beta : \mathbb{N}^3 \to \mathbb{N} \) satisfying for every sequence \( (a_0, \ldots, a_r) \) over \( \mathbb{N} \) there exist \( s, t \in \mathbb{N} \) such that for all \( i \leq r \), \( \beta(s, t, i) = a_i \).*

Now, we define \((x)_i\) to be

\[
(x)_i = \beta(L(x), R(x), i).
\]

Then, for any finite sequence \( (a_0, \ldots, a_r) \) of natural numbers, we can use one natural number \( c \) as the code such that \((c)_i = a_i \) for each \( i \leq r \). Obviously, \((x)_i\) is a recursive function, which is arithmetical.
3. Hoare’s logic based on \( PA \)

The logical language \( L' \) for Hoare’s logic with respect to Peano’s arithmetic consists of the language \( L \) of arithmetic, in which assertions describing a program’s behavior are specified and the expressions forming the right-hand sides of assignment statements and quantifier-free boolean expressions of conditionals and iterative statements are specified, plus program constructs:

- constant symbols: \( 0, 1 \);
- function symbols: \(+, \cdot\);
- binary predicate symbol: \(<\);
- variables for numbers: \( x_0, x_1, x_2, \ldots \);
- logical connectives and quantifiers: \( \neg, \rightarrow, \forall \);
- program constructs: \( :=, ;, \text{if, then, else, fi, while, do, od} \).

An expression \( e \) is defined as follows:

\[
e ::= 0 | 1 | x | e_1 + e_2 | e_1 \cdot e_2;
\]

and a boolean expression \( b \) is defined as follows:

\[
b ::= e_1 < e_2 | \neg b_1 | b_1 \rightarrow b_2.
\]

A program \( S \) is defined as follows:

\[
S ::= x ::= e | S_1; S_2 | \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi } | \text{while } b \text{ do } S_0 \text{ od}.
\]

An assertion \( p \) is defined as follows:

\[
p ::= b | \neg p_1 | p_1 \rightarrow p_2 | \forall x p_1(x).
\]

Then, an asserted program \( \varphi \) is a Hoare’s triple of the following form:

\[
\varphi ::= \{p_1\} S \{p_2\}.
\]

In the present context, we obviously prefer the more suggestive term specification to theory.
Hoare’s logic with respect to the specification $PA$, has the following axioms and proof rules for manipulating asserted programs: Let $S, S_0, S_1, S_2 \in WP; p, q, p_1, q_1, r \in L; b \in L$, a quantifier-free formula.

(1) Assignment axiom scheme: for $e$ an expression and $x$ a variable of $L'$, the asserted program

$$\{ p(e/ x) \} x := e \{ p \}$$

is an axiom, where $p(e/ x)$ stands for the result of substituting $e$ for free occurrences of $x$ in $p$.

(2) Composition rule:

$$\{ p \} S_1 \{ r \}, \{ r \} S_2 \{ q \} \Rightarrow \{ p \} S_1; S_2 \{ q \}$$

(3) Conditional rule:

$$\{ p \land b \} S_1 \{ q \}, \{ p \land \neg b \} S_2 \{ q \} \Rightarrow \{ p \} \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi} \{ q \}$$

(4) Iteration rule:

$$\{ p \land b \} S_0 \{ p \} \Rightarrow \{ p \} \text{while } b \text{ do } S_0 \text{ od} \{ p \land \neg b \}$$

(5) Consequence rule:

$$p \Rightarrow p_1, \{ p_1 \} S \{ q_1 \}, q_1 \Rightarrow q \Rightarrow \{ p \} S \{ q \}$$

And, in connection with (5),

(6) Specification axiom: Each theorem of $PA$ is an axiom.

Note that the purpose of the specification is to generate the assertions about the data types necessary to govern Consequence rule. The set of asserted programs derivable from these axioms by the proof rules we denote $HL(PA)$ and we write $HL(PA) \vdash \{ p \} S \{ q \}$ in place of $\{ p \} S \{ q \} \in HL(PA)$.

The semantics for Hoare’s logic is a Kripke’s possible world semantics, where the possible worlds are stores, and $S$ are interpreted as modalities.

Let $M = (M, I)$ be a model of $PA$ and $W$ the set of all the assignments(stores), functions from variables to $M$.

A model $M'$ for Hoare’s logic based on $PA$ is the triple $(W, \{ R_S : S \in WP \}, M)$ where $R_S$ is a binary relation on $W$ such that for any $w, w' \in W$,
• \((w, w') \in R_{x=e}\) iff \(w' = w(e^{I,w}/x)\), where \(e^{I,w}\) is defined in the following:

• \((w, w') \in R_{S_1; S_2}\) iff \((w, w') \in R_{S_1} \circ R_{S_2}\), where \((z, z') \in R_1 \circ R_2\) iff there is a \(z''\) such that \((z, z'') \in R_1\) and \((z'', z') \in R_2\);

• \((w, w') \in R_{\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}}\) iff either \(M, w \models b\) and \((w, w') \in R_{S_1}\) or \(M, w \not\models b\) and \((w, w') \in R_{S_2}\);

• \((w, w') \in R_{\text{while } b \text{ do } S_0 \text{ od}}\) iff there exist natural number \(i\) and \(w_0 = w, w_1, \ldots, w_i = w'\) such that for each \(0 \leq j < i\), \(M, w_j \models b\) and \((w_j, w_{j+1}) \in R_{S_0}\), and \(M, w' \not\models b\).

Given a possible world \(w\) and an expression \(e\), the interpretation \(e^{I,w}\) of \(e\) in \(w\) is defined as follows:

\[
e^{I,w} = \begin{cases} 0^M & \text{if } e = 0 \\ 1^M & \text{if } e = 1 \\ w(x) & \text{if } e = x \\ e^{I,w}_1 + e^{I,w}_2 & \text{if } e = e_1 + e_2 \\ e^{I,w}_1 \cdot e^{I,w}_2 & \text{if } e = e_1 \cdot e_2. \end{cases}
\]

Given a possible world \(w\) and a boolean expression \(b\), \(b\) is satisfied in \(w\), denoted by \(M, w \models b\), if

\[
\begin{array}{l}
e^{I,w}_1 <^M e^{I,w}_2 & \text{if } b = e_1 < e_2 \\
M, w \not\models b_1 & \text{if } b = \neg b_1 \\
M, w \models b_1 \Rightarrow M, w \models b_2 & \text{if } b = b_1 \rightarrow b_2,
\end{array}
\]

where instead of using \(\Rightarrow\) as the logical implication in Hoare’s logic, we use \(\rightarrow\).

Given a possible world \(w\) and an assertion \(p\), we say that \(p\) is satisfied in \(w\), denoted by \(M, w \models p\), if

\[
\begin{array}{l}
M, w \models b & \text{if } p = b \\
M, w \not\models p_1 & \text{if } p = \neg p_1 \\
M, w \models p_1 \Rightarrow M, w \models p_2 & \text{if } p = p_1 \rightarrow p_2 \\
A a \in M(M, w(a/x) \models p_1(x)) & \text{if } p = \forall xp_1(x).
\end{array}
\]

\(^1\)In syntax, we use \(\neg, \land, \rightarrow, \forall, \exists\) to denote the logical connectives and quantifiers; and in semantics we use \(\sim, \&, \Rightarrow, A, E\) to denote the corresponding connectives and quantifiers.
Given a possible world $w$, an asserted program $\{p\}S\{q\}$ is satisfied at $w$, denoted by $M, w \models \{p\}S\{q\}$, if

$$M, w \models p \Rightarrow Aw'(\langle w, w' \rangle \in R_S \Rightarrow M, w' \models q).$$

Then, $M \models \{p\}S\{q\}$ iff for any possible world $w$, $M, w \models \{p\}S\{q\}$. $PA \models \{p\}S\{q\}$ iff for any model $M$ of $PA$, $M \models \{p\}S\{q\}$. We also denote $PA \models \{p\}S\{q\}$ by $HL(PA) \models \{p\}S\{q\}$.

4. Proof of the Separation Theorem

4.1. Arithmetical definability of $\mathbb{N}$-computable functions

Let $\mathbb{N}' = (W, \{R_S : S \in WP\}, \mathbb{N})$ be the model of Hoare’s logic based on $PA$, where $R_S$ is the accessibility relation for $S \in WP$ over the standard structure $\mathbb{N}$, and $\vec{x}$ the list of variables of length $n$ occurring in while-program $S$ in the following. $R_S$ induces a vectorial function $f_S : N^n \to N^n$ such that for any $\vec{a}, \vec{c} \in N^n$,

$$f_S(\vec{a}) = \vec{c}$$

if and only if there exist $w, w' \in W$ such that $w(\vec{x}) = \vec{a}$, $w'(\vec{x}) = \vec{c}$ and $(w, w') \in R_S$. Let $f_S = (f_S^{(1)}, \ldots, f_S^{(i)}, \ldots, f_S^{(n)})$. Then the $i$th component $f_S^{(i)}$ of $f_S$ is a function from $N^n$ to $N$. We call the function $f_S^{(i)}$ is $\mathbb{N}$-computable.

Lemma 4.1.1. $f_S$ is well-defined. That is, for any $w_1, w'_1, w_2, w'_2 \in W$ with $(w_1, w'_1), (w_2, w'_2) \in R_S$, if $w_1(\vec{x}) = w_2(\vec{x})$, then $w'_1(\vec{x}) = w'_2(\vec{x})$.

Proof. By induction on the structure of $S$. \hfill \square

Next, we study how to arithmetically define $f_S$. Construct the formula $\alpha_S$ inductively in $L$.

Assignment: $S \equiv x_i := e$

$$\alpha_S(\vec{x}, \vec{y}) \equiv y_i = e(\vec{x}) \wedge \bigwedge_{j \neq i} y_j = x_j;$$

Composition: $S \equiv S_1; S_2$

$$\alpha_S(\vec{x}, \vec{y}) \equiv \exists \vec{z} (\alpha_{S_1}(\vec{x}, \vec{z} / \vec{y}) \wedge \alpha_{S_2}(\vec{z} / \vec{x}, \vec{y}));$$
Conditional: \( S \equiv if \ b \ then \ S_1 \ else \ S_2 \ fi \)

\[
\alpha_S(\vec{x}, \vec{y}) \equiv (b(\vec{x}) \land \alpha_{S_1}(\vec{x}, \vec{y})) \lor (\neg b(\vec{x}) \land \alpha_{S_2}(\vec{x}, \vec{y}));
\]

Iteration: \( S \equiv while \ b \ do \ S_0 \ od \). First, define

\[
A_S(i, w, \vec{x}, \vec{y}) \equiv \vec{x} = \overline{(w)_0} \land \forall j < i(b((w)_j/ \vec{x}) \land \alpha_{S_0}((w)_j/\vec{x}, (w)_{j+1}/ \vec{y}) \land \vec{y} = (w)_i)
\]

then, set

\[
\alpha^*_S(i, \vec{x}, \vec{y}) \equiv \exists w A_S(i, w, \vec{x}, \vec{y})
\]

and so define

\[
\alpha_S(\vec{x}, \vec{y}) \equiv \exists i \alpha^*_S(i, \vec{x}, \vec{y}) \land \neg b(\vec{y}/\vec{x})
\]

Lemma 4.1.2. \( f_S \) is arithmetically definable by \( \alpha_S \).

Proof. It is equivalent to prove that for any \( \vec{a}, \vec{c} \in N^n, f_S(\vec{a}) = \vec{c} \Leftrightarrow \alpha_S(\vec{a}, \vec{c}) \) is correct.

The argument is an induction on \( S \) for which the basis is the assignment statement.

Assignment: \( S \equiv x_i := e \). \( f_S(\vec{a}) = \vec{c} \Leftrightarrow (v(\vec{a}/\vec{x}), v(\vec{c}/\vec{x})) \in R_S \Leftrightarrow c_i = e^{t,v(\vec{a}/\vec{x})} \) and \( c_j = a_j \) for \( j \neq i \Leftrightarrow c_i = e(a) \land \bigwedge_{j \neq i} c_j = a_j \) is correct \( \Leftrightarrow \alpha_S(\vec{a}, \vec{c}) \) is correct.

Composition: \( S \equiv S_1; S_2 \). \( f_S(\vec{a}) = \vec{c} \Leftrightarrow (v(\vec{a}/\vec{x}), v(\vec{c}/\vec{x})) \in R_S \Leftrightarrow (\Rightarrow \text{There exists a vector } \vec{d} \text{ of natural numbers such that } (v(\vec{a}/\vec{x}), v(\vec{d}/\vec{x})) \in R_{S_1} \land (v(\vec{d}/\vec{x}), v(\vec{c}/\vec{x})) \in R_{S_2}) \Leftrightarrow f_{S_1}(\vec{a}) = \vec{d} \land f_{S_2}(\vec{d}) = \vec{c} \Leftrightarrow (\Leftarrow \text{There exists a vector } \vec{d} \text{ of natural numbers such that } \alpha_{S_1}(\vec{a}, \vec{d}) \text{ is correct and } \alpha_{S_2}(\vec{d}, \vec{c}) \text{ is correct } \Leftrightarrow \exists z \ (\alpha_{S_1}(\vec{a}, z/\vec{y}) \land \alpha_{S_2}(z/\vec{x}, \vec{c})) \text{ is correct } \Leftrightarrow \alpha_S(\vec{a}, \vec{c}) \text{ is correct})

Conditional: \( S \equiv if \ b \ then \ S_1 \ else \ S_2 \ fi \). \( f_S(\vec{a}) = \vec{c} \Leftrightarrow (v(\vec{a}/\vec{x}), v(\vec{c}/\vec{x})) \in R_S \Leftrightarrow \exists b \ (v(\vec{a}/\vec{x}), v(\vec{c}/\vec{x})) \in R_{S_1} \lor \exists b \ (v(\vec{a}/\vec{x}), v(\vec{c}/\vec{x})) \in R_{S_2} \Leftrightarrow \exists b \ (\neg b(\vec{a}) \land \alpha_{S_2}(\vec{a}, \vec{c}) \text{ is correct}) \lor (b(\vec{a}) \land \alpha_{S_1}(\vec{a}, \vec{c}) \text{ is correct}) \Leftrightarrow \alpha_S(\vec{a}, \vec{c}) \text{ is correct} \Leftrightarrow \exists b (b(\vec{a}) \land \alpha_{S_1}(\vec{a}, \vec{c})) \lor (b(\vec{a}) \land \alpha_{S_2}(\vec{a}, \vec{c})) \text{ is correct} \Leftrightarrow \alpha_S(\vec{a}, \vec{c}) \text{ is correct}


Iteration: $S \equiv$ while $b$ do $S_0$ od. $f_S(a) = a \Rightarrow (v(a / x), v(c / x)) \in R_S$ \[\iff\] there exist natural number $i$ and vectors of natural numbers $a_0, \ldots, a_i$ such that $\hat{a} = \hat{a}_0$, for any $j$ less than $i$, $\mathbf{N}, v(a_j / x) \models b$ and $(v(a_j / x), v(a_{j+1} / x)) \in R_{S_0}$, $\hat{c} = \hat{a}_i$, $\mathbf{N}, v(c / x) \not\models b$ \[\iff\] there exist natural number $i$ and vectors of natural numbers $a_0, \ldots, a_i$ such that $\hat{a} = \hat{a}_0$, for any $j$ less than $i$, $\mathbf{N} \models b(\hat{a}_j)$ and $f_{S_0}(\hat{a}_j) = a_{j+1}$, $\hat{c} = \hat{a}_i$, $\mathbf{N} \models \neg b(\hat{c})$ \[\iff\] there exist natural number $i$ and vectors of natural numbers $a_0, \ldots, a_i$ such that $\hat{a} = \hat{a}_0$, for any $j$ less than $i$, $b(\hat{a}_j)$ is correct and $\alpha_{S_0}(\hat{a}_j, \hat{a}_{j+1})$ is correct, $\hat{c} = \hat{a}_i$, $\neg b(\hat{c})$ is correct \[\iff\] there exist natural number $i$ and vectors of natural numbers $a_0, \ldots, a_i$ such that $\hat{a} = \hat{a}_0$ \[\land\] $\forall j < i \left( b(\hat{a}_j) \land \alpha_{S_0}(\hat{a}_j, \hat{a}_{j+1}) \right)$ \[\land\] $\neg b(\hat{c})$ is correct \[\iff\] there exist natural numbers $i$ and $w$ such that $\hat{a} = (\hat{w})_0$ \[\land\] $\forall j < i \left( b(\hat{w}_j) \land \alpha_{S_0}(\hat{w}_j, (\hat{w})_{j+1}) \right)$ \[\land\] $\neg b(\hat{c})$ is correct \[\iff\] there exist natural numbers $i$ and $w$ such that $A_S(i, w, \hat{a}, \hat{c}) \land \neg b(\hat{c})$ is correct \[\iff\] $\exists i \exists w B(i, w, \hat{a}, \hat{c}) \land \neg b(\hat{c})$ is correct \[\iff\] $\alpha_S(\hat{a}, \hat{c})$ is correct.

\[\Box\]

Define $\alpha_S^{(i)}(\hat{x}, y)$ as follows:

$\alpha_S^{(i)}(\hat{x}, y) := \exists y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n \alpha_S(\hat{x}, (y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_n))$.

Then, we can easily achieve that $f_S^{(i)}$ is arithmetically defined by $\alpha_S^{(i)}$ from the above lemma. Actually, these two definabilities are interconvertible.

### 4.2. The definition of $SP(p, S)$

Now, we define strongest postconditions. Given the model $\mathbf{N}' = (W, \{R_S : S \in WP\}, \mathbf{N})$ of Hoare’s logic based on $PA$, $p \in L$ and $S \in WP$, then the strongest postcondition $sp_{\mathbf{N}}(p, S)$ of $p$ and $S$ relative to $\mathbf{N}$ is the set of possible worlds:

$\{ w \in W : Ew' \in W(\mathbf{N}, w' \models p \& (w', w) \in R_S) \}$.

From the definitions, it immediately follows that

**Lemma 4.2.1.** $\mathbf{N} \models \{p\} S(q) \iff sp_{\mathbf{N}}(p, S) \subseteq \{ w \in W : \mathbf{N}, w \models q \}$.  

13
Like the definition of $SP(p, S)$ in [11, p271], we can inductively define $SP(p, S)$ over the structure of $S$ as follows:

**Assignment:** $S \equiv x_i := e$

$$SP(p, S)(\vec{x}) \equiv \exists \ u \ (p(\vec{u} / \vec{x}) \land x_i = e(\vec{u} / \vec{x}) \land \bigwedge_{j \neq i} x_j = u_j);$$

**Composition:** $S \equiv S_1 ; S_2$

$$SP(p, S)(\vec{x}) \equiv SP(SP(p, S_1), S_2)(\vec{x});$$

**Conditional:** $S \equiv if \ b \ then \ S_1 \ else \ S_2 \ fi$

$$SP(p, S)(\vec{x}) \equiv SP(p \land b, S_1)(\vec{x}) \lor SP(p \land \neg b, S_2)(\vec{x});$$

**Iteration:** $S \equiv while \ b \ do \ S_0 \ od$

$$SP(p, S)(\vec{x}) \equiv INV(p, b, S_0)(\vec{x}) \land \neg b(\vec{x}),$$

where $INV(p, b, S_0)$ is the loop invariant formula (refer to Invariant Law in [11, p277]) built up as follows.

First, define

$$B(p, b, S_0)(i, w, \vec{x}) \equiv p((w)_0 / \vec{x}) \land \forall j < i \ SP(\vec{x} = (w)_j \land b, S_0)((w)_{j+1} / \vec{x}) \land \vec{x} = (w)_i.$$  

Next set

$$INV^*(p, b, S_0)(i, \vec{x}) \equiv \exists w B(p, b, S_0)(i, w, \vec{x})$$

and so define

$$INV(p, b, S_0)(\vec{x}) \equiv \exists i INV^*(p, b, S_0)(i, \vec{x}).$$

The definition of $SP(p, S)$ in [11, p271] entails:

**Lemma 4.2.2.** $SP(p, S)$ arithmetically defines $SP_N(p, S)$.

From the preceding two lemmas, it deduces that

**Lemma 4.2.3.** $N \models \{p\} S\{q\} \iff N \models SP(p, S) \rightarrow q.$

Lemma 4.2.3 is a special case of statement (3) in Theorem 1.1, when restricting the model of $PA$ to $N$.  

14
4.3. Proof of the Separation Theorem: The equivalent form of $SP(p, S)$ in $PA$

Until now, we have finished all the preparations for the Separation Theorem. Let’s turn to the last part of the proof of the theorem. In the sequel, we argue that $SP(p, S)$ has an equivalent form $\exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$ in $PA$.

**Theorem 4.3.1.** $PA \vdash SP(p, S)(\vec{x}) \leftrightarrow \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$.

**Proof.** By Gödel’s Soundness and Completeness Theorem,

$PA \vdash SP(p, S)(\vec{x}) \leftrightarrow \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$

if and only if

$PA \models SP(p, S)(\vec{x}) \leftrightarrow \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$.

Consequently, it suffices to prove for any $M \models PA$,

$M \models SP(p, S)(\vec{x}) \leftrightarrow \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$.

Fix $M \models PA$. The argument is by induction on the structure of $S$ for which the basis is the assignment.

**Assignment:** $S \equiv x_i := e$. According to the definition, $SP(p, S)(\vec{x})$ is equivalent in $M$ to

$\exists \vec{u} (p(\vec{u} / \vec{x}) \land x_i = e(\vec{u} / \vec{x}) \land \bigwedge_{j \neq i} x_j = u_j)$,

by pure logic and the definition of $\alpha_S(\vec{x}, \vec{y})$, it is equivalent to

$\exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$.

Thus,

$M \models SP(p, S)(\vec{x}) \leftrightarrow \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$.

**Composition:** $S \equiv S_1; S_2$. The induction hypothesis applied to $S_1$ and $S_2$ yields

$M \models SP(p, S_1)(\vec{x}) \leftrightarrow \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_{S_1}(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$,

$M \models SP(S(p, S_1), S_2)(\vec{x}) \leftrightarrow \exists \vec{u} (SP(p, S_1)(\vec{u} / \vec{x}) \land \alpha_{S_2}(\vec{u} / \vec{x}, \vec{x} / \vec{y}))$.
Consider this last formula through several transformations: replacing \( \vec{u} \) by \( \vec{z} \), it is equivalent in \( M \) to

\[
\exists \vec{z} \left( SP(p, S_1)(\vec{z} / \vec{x}) \land \alpha_{S_2}(\vec{z} / \vec{x}, \vec{x} / \vec{y}) \right).
\]

With the substitution of \( \exists \vec{u} \left( p(\vec{u} / \vec{x}) \land \alpha_{S_1}(\vec{u} / \vec{x}, \vec{z} / \vec{y}) \right) \) for \( SP(p, S_1)(\vec{z} / \vec{x}) \), it is equivalent to

\[
\exists \vec{z} \left( \exists \vec{u} \left( p(\vec{u} / \vec{x}) \land \alpha_{S_1}(\vec{u} / \vec{x}, \vec{z} / \vec{y}) \right) \land \alpha_{S_2}(\vec{z} / \vec{x}, \vec{x} / \vec{y}) \right).
\]

By pure logic, it is equivalent to

\[
\exists \vec{u} \left( p(\vec{u} / \vec{x}) \land \exists \vec{z} \left( \alpha_{S_1}(\vec{u} / \vec{x}, \vec{z} / \vec{y}) \land \alpha_{S_2}(\vec{z} / \vec{x}, \vec{x} / \vec{y}) \right) \right).
\]

By the definition of \( \alpha_S(\vec{x}, \vec{y}) \), it is equivalent to

\[
\exists \vec{u} \left( p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}) \right).
\]

Thus, by the definition, \( SP(p, S)(\vec{x}) \) is equivalent in \( M \) to

\[
SP(SP(p, S_1), S_2)(\vec{x}),
\]

By the above argument, it is equivalent to

\[
\exists \vec{u} \left( p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}) \right).
\]

Therefore,

\[
M \models SP(p, S)(\vec{x}) \iff \exists \vec{u} \left( p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y}) \right).
\]

Conditional: \( S \equiv \text{if } b \text{ then } S_1 \text{ else } S_2 \). By the induction hypothesis applied to \( S_1, S_2 \), it yields

\[
M \models SP(p \land b, S_1)(\vec{x})
\]

\[
\iff \exists \vec{u} \left( p(\vec{u} / \vec{x}) \land b(\vec{u} / \vec{x}) \land \alpha_{S_1}(\vec{u} / \vec{x}, \vec{x} / \vec{y}) \right),
\]

\[
M \models SP(p \land \neg b, S_2)(\vec{x})
\]

\[
\iff \exists \vec{u} \left( p(\vec{u} / \vec{x}) \land \neg b(\vec{u} / \vec{x}) \land \alpha_{S_2}(\vec{u} / \vec{x}, \vec{x} / \vec{y}) \right).
\]
According to the definition, $SP(p, S)(\overrightarrow{x})$ is equivalent in $M$ to

$$SP(p \land b, S_1)(\overrightarrow{x}) \lor SP(p \land \neg b, S_2)(\overrightarrow{x})$$

Consider this last formula through several transformations: by substitutions, it is equivalent in $M$ to

$$\exists \overrightarrow{u} \left( p(\overrightarrow{u} / \overrightarrow{x}) \land b(\overrightarrow{u} / \overrightarrow{x}) \land \alpha S_1(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right) \lor \exists \overrightarrow{u} \left( p(\overrightarrow{u} / \overrightarrow{x}) \land \neg b(\overrightarrow{u} / \overrightarrow{x}) \land \alpha S_2(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right).$$

By pure logic, it is equivalent to

$$\exists \overrightarrow{u} \left( p(\overrightarrow{u} / \overrightarrow{x}) \land \left( (b(\overrightarrow{u} / \overrightarrow{x}) \land \alpha S_1(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right) \lor \left( \neg b(\overrightarrow{u} / \overrightarrow{x}) \land \alpha S_2(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right) \right).$$

By the definition of $\alpha_S(\overrightarrow{x}, \overrightarrow{y})$, it is equivalent to

$$\exists \overrightarrow{u} \left( p(\overrightarrow{u} / \overrightarrow{x}) \land \alpha_S(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right).$$

So, we obtain

$$M \models SP(p, S)(\overrightarrow{x}) \leftrightarrow \exists \overrightarrow{u} \left( p(\overrightarrow{u} / \overrightarrow{x}) \land \alpha_S(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right).$$

Iteration: $S \equiv \text{while } b \text{ do } S_0 \text{ od}$. Applying the induction hypothesis to $S_0$ yields

$$M \models SP(\overrightarrow{x} = (w)_j \land b, S_0)(\overrightarrow{x}) \leftrightarrow \exists \overrightarrow{u} \left( \overrightarrow{u} = (w)_j \land b(\overrightarrow{u} / \overrightarrow{x}) \land \alpha S_0(\overrightarrow{u} / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}) \right).$$

By the generalized coding function assumed by Bergstra, et.al[11], the last formula in the above theorem is equivalent in $M$ to

$$b((w)_j / \overrightarrow{x}) \land \alpha S_0((w)_j / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}).$$

Then,

$$M \models SP(\overrightarrow{x} = (w)_j \land b, S_0)(\overrightarrow{x}) \leftrightarrow b((w)_j / \overrightarrow{x}) \land \alpha S_0((w)_j / \overrightarrow{x}, \overrightarrow{x} / \overrightarrow{y}).$$
Substituting \((w)_{j+1}\) for each free occurrence of \(\vec{x}\) in the above formula gives

\[
\begin{align*}
\mathbf{M} & \models SP \left( \vec{x} = (w)_j \land b, S_0 \left( (w)_{j+1} / \vec{x} \right) \leftrightarrow b((w)_j / \vec{x}) \land \alpha_{S_0}(w)_j/\vec{x}, (w)_{j+1}/\vec{y} \right). 
\end{align*}
\]

From the definition, it follows directly that \(B(p, b, S_0)(i, w, \vec{x})\) is equivalent in \(\mathbf{M}\) to

\[
\begin{align*}
p((w)_0 / \vec{x}) \land \forall j < i SP \left( \vec{x} = (w)_j \land b, S_0 \left( (w)_{j+1} / \vec{x} \right) \right) \\
\land \vec{x} = (w)_i.
\end{align*}
\]

By the above formula, it is equivalent to

\[
\begin{align*}
p((w)_0 / \vec{x}) \land \forall j < i \left( b((w)_j / \vec{x}) \right) \\
\land \alpha_{S_0}(w)_j/\vec{x}, (w)_{j+1}/\vec{y}) \land \vec{x} = (w)_i.
\end{align*}
\]

According to the definition of \(A_S(i, w, \vec{x}, \vec{y})\), it is equivalent to

\[
\begin{align*}
p((w)_0 / \vec{x}) \land A_S(i, w, (w)_0/\vec{x}, \vec{x} / \vec{y}).
\end{align*}
\]

With the generalized coding function, it is equivalent in \(\mathbf{M}\) to

\[
\begin{align*}
\exists \vec{u} \left( p(\vec{u} / \vec{x}) \land A_S(i, w, \vec{u} / \vec{x}, \vec{x} / \vec{y}) \right).
\end{align*}
\]

Therefore,

\[
\begin{align*}
\mathbf{M} & \models B(p, b, S_0)(i, w, \vec{x}) \\
& \leftrightarrow \exists \vec{u} \left( p(\vec{u} / \vec{x}) \land A_S(i, w, \vec{u} / \vec{x}, \vec{x} / \vec{y}) \right).
\end{align*}
\]

By the definition, \(SP(p, S)(\vec{x})\) is equivalent in \(\mathbf{M}\) to

\[
\begin{align*}
INV(p, b, S_0)(\vec{x}) \land \neg b(\vec{x}).
\end{align*}
\]

Now, consider the last formula through several transformations: it is equivalent in \(\mathbf{M}\) to

\[
\begin{align*}
\exists i \exists w B(p, b, S_0)(i, w, \vec{x}) \land \neg b(\vec{x}).
\end{align*}
\]
By the above formula, it is equivalent to
\[ \exists i \exists w \exists \vec{u} (p(\vec{u} / \vec{x}) \land A_S(i, w, \vec{u} / \vec{x}, \vec{x} / \vec{y})) \land \neg b(\vec{x}). \]

By pure logic, it is equivalent to
\[ \exists \vec{u} (p(\vec{u} / \vec{x}) \land \exists i \exists w A_S(i, w, \vec{u} / \vec{x}, \vec{x} / \vec{y}) \land \neg b(\vec{x})). \]

And, clearly, by the definition of \( \alpha_S(\vec{x}, \vec{y}) \), this last formula is equivalent in \( M \) to
\[ \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y})). \]

Thus,
\[ M \models SP(p, S)(\vec{x}) \iff \exists \vec{u} (p(\vec{u} / \vec{x}) \land \alpha_S(\vec{u} / \vec{x}, \vec{x} / \vec{y})). \]

And we are done.

5. Discussion and Future Work

The corollary \[1.3\] looks on \( \alpha_S \) as the logical characterization of while-program \( S \) in \( PA \). In the sequel, we shall extend the computability induced by \( S \) from the standard structure to any nonstandard model of \( PA \) through \( \alpha_S \). Lemma \[4.1.2\] presents that \( \alpha_S \) arithmetically defines \( f_N^S \), i.e. for any vectors of natural numbers \( \vec{a}, \vec{c}, f_N^S(\vec{a}) = \vec{c} \iff N \models \alpha_S(\vec{a}, \vec{c}) \). Then for any nonstandard model \( M = (M, I) \) of \( PA \), we define \( f_M^S \) such that for any elements \( \vec{a}, \vec{c} \in M^n, f_M^S(\vec{a}) = \vec{c} \iff M \models \alpha_S(\vec{a}, \vec{c}) \). Thus, \( f_M^S \) is an extension of \( f_N^S \) with the uniform \( \Sigma_1 \)-definability in \( L \). It remains to verify the validity of this underlying semantics(type-1 semantics).

Consider the while-program \( S \) as follows:
\[ y := 0; \text{while } y < x \text{ do } y := y + 1 \text{ od}, \]
where the program variable \( x \) is the only input and output variable. Then, by the generalized coding function, \( f_N^S = id_N \) and \( \alpha_S(x, y) \) is logically equivalent to \( y = x \). By \( id_N \) we mean the identity function defined in \( N \), i.e. \( \text{dom}(id_N) = N \). Let \( M = (M, I) \) be any nonstandard model of \( PA \). Then, \( M \)-computable function \( f_M^S \) defined in \( M \) by \( \alpha_S \), in which the generalized coding function codes the \( M \)-finite sequence of pairs \( (0, x), (1, x) \ldots (x, x) \), is
id^M(dom(id^M) = M). Therefore, the type-1 semantics extends the number of halting steps from N-finiteness to M-finiteness, which makes L expressive relative to M.

On the other hand, up to now, we have known that the number of iterative steps of a loop is defined in the standard part of M, as treated in this paper. In this semantics(type-2 semantics), the M-computable function induced by S on M is defined in N^M, i.e. id^{N^M}. By the Overspill principle[12, p309], N^M is not arithmetically defined in M. Hence, id^{N^M} is not arithmetically defined in M.

To make the type-2 semantics defined logically, introduce a new predicate symbol to the language of arithmetic, which defines N^M in M, and restrict its occurrences in formulas, say in a guarded fashion. The paper [15] is an attempt for the type-2 semantics.

Acknowledgement

I gratefully thank Prof. Yuefei Sui for many long and interesting discussions about Hoare’s logic, and for review of this paper. I also thank Prof. Wenhui Zhang for comments on earlier versions of this paper.

References

[1] K. R. Apt, Ten year’s of Hoare’s logic: a survey, ACM Trans. Programming Languages and Systems 3(4) (1981) 431-483.

[2] C. A. R. Hoare, An axiomatic basis for computer programming, Comm. ACM 12 (1969) 576-580.

[3] S. A. Cook, Soundness and completeness of an axiom system for program verification, SIAM J. Comput. 7 (1978) 70-90.

[4] R. J. Lipton, A necessary and sufficient condition for the existence of Hoare Logics, In Proc. 18th IEEE Symp. Foundations of Computer Science (1977) 1-6.

[5] M. Wand, A new incompleteness result for Hoare’s system, J. ACM 25(1) (1978) 168-175.
[6] E. M. Clarke JR., Programming language constructs for which it is impossible to obtain good Hoare axiom systems, J. ACM 26(1) (1979) 129-147.

[7] J. C. Reynolds, Separation logic: a logic for shared mutable data structures, LICS (2002) 55-74.

[8] P. W. O’Hearn, H. Yang, J. C. Reynolds, Separation and information hiding, in Proceedings of POPL (2004) 268-280.

[9] M. Parkinson, G. Bierman, Separation logic and abstraction, in Proceedings of POPL (2005) 247-258.

[10] G. E. Hughes, M. J. Cresswell, A new introduction to modal logic, Routledge, London, 1996.

[11] J. A. Bergstra, J. V. Tucker, Hoare’s logic and peano’s arithmetic, Theoretical Computer Science 22 (1983) 265-284.

[12] G. S. Boolos, J. P. Burgess, R. C. Jeffrey, Computability and logic, fifth edition, Cambridge University Press, 2007.

[13] Martin Davis, Computability & unsolvability, Courier Dover Publications, New York, 1982.

[14] H. -D. Ebbinghaus, J. Flum, W. Thomas, Mathematical logic, second ed., Springer, New York, 1994.

[15] C. Cao, Y. Sui, Z. Zhang, The M-computations induced by accessibility relations in nonstandard models M of Hoare logic, Institute of Computing Technology, Chinese Academy of Sciences, 2013.

[16] K. Hrbacek, T. Jech, Introduction to Set Theory, third ed., Marcel Dekker, New York, 1999.

[17] H. Rogers, Theory of Recursive Functions and Effective Computability, The MIT Press, 1987.

[18] J. Barwise, Handbook of Mathematical Logic, North-Holland, Amsterdam, 1977.