Program Algebras with Monotone Floyd-Hoare Composition

Andrii Kryvolap[1], Mykola Nikitchenko[1] and Wolfgang Schreiner[2]

[1] Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

krivolapa@gmail.com, nikitchenko@unicyb.kiev.ua

[2] Johannes Kepler University, Linz, Austria

Wolfgang.Schreiner@risc.jku.at

Abstract. In the paper special program algebras of partial predicates and functions are described. Such algebras form a semantic component of a modified Floyd-Hoare logic constructed on the base of a composition-nominative approach. According to this approach, Floyd-Hoare assertions are presented with the help of a special composition called Floyd-Hoare composition. Monotonicity and continuity of this composition are proved. The language of the modified Floyd-Hoare logic is described. Further, the inference rules for such logic are studied, their soundness conditions are specified. The logic constructed can be used for program verification.

Keywords. Program algebra, program logic, composition-nominative approach, partial predicate, soundness

Key terms. FormalMethod, VerificationProcess

1 Introduction

Program logics are the main formalisms used for proving assertions about program properties. A well-known Floyd-Hoare logic [1, 2] is an example of such logics. Semantically, this logic is defined for a case of total predicates and functions though programs can be partial. In this case assertions can be presented with the help of a special composition over total predicates and functions called Floyd-Hoare composition (FH-composition). However, a straightforward extension of classical Floyd-Hoare logic for partial predicates and functions meets some difficulties. The first one is that the classical FH-composition will not be monotone. Monotonicity means that the result of the mapping evaluation remains the same on extended data, if it was evaluated on the initial data. This important property grants the possibility to reason about the correctness of the program based on the correctness of its approximations.

That is why the need of a modified definition of the classical Floyd-Hoare logic for the case of partial mappings arises. Here we will consider only mappings (predicates, ordinary functions, and program functions) defined over flat nominative data (nominative sets). Such data are treated as collections of named values. Mappings over such data are called quasiary mappings [3]. The obtained program algebras are called quasiary program algebras. They form a semantic component of quasiary Floyd-Hoare logics.

The syntactic component of such logics is presented by their languages and systems of inference rules. We study the possibility to use classical rules for modified logics with a monotone Floyd-Hoare composition. Systems of such inference rules should be sound and complete to be of a practical use. This could be achieved by adding proper restrictions to the inference rules of the classical Floyd-Hoare logic that fail to be correct. It should be also shown that by weakening additional restrictions we obtain a system of the inference rules that is not sound. This will prove that restrictions are necessary.

The rest of the paper is structured as follows. In Section 2 we describe program algebras of quasiary predicates and functions on different levels of abstraction, define a modified Floyd-Hoare composition and specify the syntax for the modified logic. In Section 3 we prove the main properties of this composition. In Section 4 we study the soundness of the system of inference rules for the introduced program algebras. Finally, we formulate conclusions in Section 5.

2 Quasiary Program Algebras

To modify the classical Floyd-Hoare logic for partial quasiary mappings, we will use semantic-syntactic scheme [3-5]. This means that we will first define the semantics in the form of classes of quasiary program algebras. Then the language of the logic will be defined as well as the interpretation mappings.

To emphasize a mapping's *partiality/totality* we write the sign $\xrightarrow{p}$ for partial mappings and the sign $\xrightarrow{t}$ for total mappings. Given an arbitrary partial mapping $\mu: D \xrightarrow{p} D'$, $d \in D$, $S \subseteq D$, $S' \subseteq D'$ we write:

- $\mu(d)\downarrow$ to denote that μ is defined on d;
- $\mu(d)\downarrow = d'$ to denote that μ is defined on d with a value d';
- $\mu(d)\uparrow$ to denote that μ is undefined on d;
- $\mu[S] = \{\mu(d) \mid \mu(d)\downarrow, d \in S\}$ to denote the image of S under μ;
- $\mu^{-1}[S'] = \{d \mid \mu(d)\downarrow, \mu(d) \in S'\}$ to denote the preimage (inverse image) of S' under μ.

2.1 Classes of quasiary mappings

Let V be a set of *names (variables)*. Let A be a set of *basic values*. Given V and A, the class ${}^{V}A$ of *nominative sets* is defined as the class of all partial mappings from V to A,

thus, ${}^{V}A=V \xrightarrow{p} A$. Informally speaking, nominative sets represent states of variables.

Though nominative sets are defined as mappings, we follow mathematical traditions and also use a set-like notation for these objects. In particular, the notation $d = [v_i \mapsto a_i \mid i \in I]$ describes a nominative set d where $v_i \mapsto a_i \in_n d$ means that $d(v_i)$ is defined and its value is a_i ($d(v_i)\downarrow=a_i$). The main operation for nominative sets is the binary *total overriding operation* ∇: ${}^{V}A \times {}^{V}A \xrightarrow{t} {}^{V}A$ defined by the formula $d_1 \nabla d_2 = [v \mapsto a \mid v \mapsto a \in_n d_2 \vee (v \mapsto a \in_n d_1 \wedge \neg \exists a'(v \mapsto a' \in_n d_2))]$. Intuitively, given d_1 and d_2 this operation yields a new nominative set which consists of named pairs of d_2 and those pairs of d_1 whose names do not occur in d_2.

Let $Bool = \{F, T\}$ be the set of Boolean values. Let $Pr^{V,A}={}^{V}A \xrightarrow{p} Bool$ be the set of all partial predicates over ${}^{V}A$. Such predicates are called *partial quasiary predicates*. Let $Fn^{V,A}={}^{V}A \xrightarrow{p} A$ be the set of all partial functions from ${}^{V}A$ to A. Such functions are called *partial quasiary ordinary functions*. Here 'ordinary' means that the range of such functions is the set of basic values A. Let $FPrg^{V,A}={}^{V}A \xrightarrow{p} {}^{V}A$ be the set of all partial functions from ${}^{V}A$ to ${}^{V}A$. Such functions are called *bi-quasiary functions*.

Quasiary predicates represent conditions which occur in programs, quasiary ordinary functions represent the semantics of program expressions, and biquasiary functions represent program semantics.

The terms 'partial' and 'ordinary' are usually omitted. In a general term, elements from $Pr^{V,A}$, $Fn^{V,A}$, and $FPrg^{V,A}$ are called *quasiary mappings*.

2.2 Hierarchy of program algebras and logics

Based on algebras with three carriers ($Pr^{V,A}$, $Fn^{V,A}$, and $FPrg^{V,A}$) we can define logics of three types:

- *Pure quasiary predicate logics based on algebras with one sort:* $Pr^{V,A}$
- *Quasiary predicate-function logics based on algebras with two sorts:* $Pr^{V,A}$ and $Fn^{V,A}$
- *Quasiary program logics based on algebras with three sorts:* $Pr^{V,A}$, $Fn^{V,A}$, and $FPrg^{V,A}$

For logics of pure quasiary predicates we identify renominative, quantifier, and quantifier-equational levels.

Renominative logics [3] are the most abstract among above-mentioned logics. The main new compositions for these logics are the compositions of renomination (renaming) of the form $R^{v_1,\dots,v_n}_{x_1,\dots,x_n} : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$. Intuitively, given a quasiary predicate p and a nominative set d the value of $R^{v_1,\dots,v_n}_{x_1,\dots,x_n}(p)(d)$ is evaluated in the following way: first, a new nominative set d' is constructed from d by changing the values of the

names $v_1,...,v_n$ in d to the values of the names $x_1,..., x_n$ respectively; then the predicate p is applied to d'. The obtained value (if it was evaluated) will be the result of $R^{v_1,...,v_n}_{x_1,...,x_n}(p)(d)$. For this composition we will also use a simplified notation $R^{\overline{v}}_{\overline{x}}$. The basic compositions of renominative logics are $\vee$, $\neg$, and $R\frac{\overline{v}}{\overline{x}}$. Note, that renomination (primarily in syntactical aspects) is widely used in classical logic, lambda-calculus, and specification languages like Z-notation, B, TLA, RAISE, ASM, etc.

At the *quantifier* level, all basic values can be used to construct different nominative sets to which quasiary predicates can be applied. This allows one to introduce the compositions of quantification of the form $\exists x$ in style of Kleene's strong quantifiers. The basic compositions of logics of the quantifier level are $\vee$, $\neg$, $R\frac{\overline{v}}{\overline{x}}$, and $\exists x$.

At the *quantifier-equational* level, new possibilities arise for equating and differentiating values with special 0-ary compositions of the form $=_{xy}$ called equality predicates. Basic compositions of logics of the quantifier-equational level are $\vee$, $\neg$, $R^{\overline{v}}_{\overline{x}}$, $\exists x$, and $=_{xy}$.

All specified logics (renominative, quantifier, and quantifier-equational) are based on algebras that have only one sort: a class of quasiary predicates.

For quasiary predicate-function logics we identify the function level and the function-equational level.

At the *function* level, we have extended capabilities for the formation of new arguments of functions and predicates. In this case it is possible to introduce the superposition compositions $S^{\overline{v}}_F$ and $S^{\overline{v}}_P$ (see [4, 5]), which formalize substitution of functions into function and predicate respectively. Also special null-ary denomination parametric compositions (functions) $'x$ are introduced. The introduction of such functions allows one to model renomination compositions with the help of superpositions. The basic compositions of logics of the function level are $\vee$, $\neg$, $S^{\overline{v}}_F$, $S^{\overline{v}}_P$, $\exists x$, and $'x$.

At the function-equational level, a special equality composition = can be introduced additionally. The basic compositions of logics of the function-equational level are $\vee$, $\neg$, $S^{\overline{v}}_F$, $S^{\overline{v}}_P$, $\exists x$, $'x$, and $=$. At this level different classes of first-order logics can be presented.

This means that two-sorted algebras (with sets of predicates and functions as sorts and above-mentioned compositions as operations) form a semantic base for first-order CNL.

The level of *program logics* is quite rich. Investigation of such logics is a special challenge; here we will study semantic properties of a modified *Floyd-Hoare logic*. To define such logics we should first define program algebras with program compositions as their operations. Such compositions correspond to the main structures of programs. In the simplest case they are:

- The parametric assignment composition $AS^x : Fn^{V,A} \to FPrg^{V,A}$
- The composition of sequential execution $\bullet : FPrg^{V,A} \times FPrg^{V,A} \to FPrg^{V,A}$
- The conditional composition $IF : Pr^{V,A} \times FPrg^{V,A} \times FPrg^{V,A} \to FPrg^{V,A}$
- The cyclic composition (loop) $WH : Pr^{V,A} \times FPrg^{V,A} \to FPrg^{V,A}$

Additionally we need compositions that describe properties of the programs. The Floyd-Hoare composition $FH : Pr^{V,A} \times FPrg^{V,A} \times Pr^{V,A} \to Pr^{V,A}$ is the most important of them. Its formal definition will be given in the next subsection.

2.3 Formal definition of a Floyd-Hoare composition

The required definition stems from the treatment of Floyd-Hoare assertions with total predicates (see, for example, [6]). Namely, an assertion $\{p\}f\{q\}$ is said to be valid if and only if

$$\text{for all } d \text{ from } {}^{V}A \text{ if } p(d) = T, f(d)\downarrow = d' \text{ for some } d' \text{ then } q(d') = T \tag{1}$$

Note, that we do not make a distinction between a formula and its interpretation. Thus, we treat, say, p as a formula in the assertion $\{p\}f\{q\}$ and as a predicate of the program algebra.

The definition (1) permits to treat $\{p\}f\{q\}$ as a predicate because this is a pointwise definition. Rewriting this definition for different cases we get the following matrices (table 1) specifying the logical values of $\{p\}f\{q\}$ for an arbitrarily d:

Table 1. Logical values of $\{p\}f\{q\}$ for total predicates.

a) $f(d)$ is defined

$p(d) \setminus q(f(d))$	F	T
F	T	T
T	F	T

b) $f(d)$ is undefined

$p(d)$	$\{p\}f\{q\}(d)$
F	T
T	T

Our aim is to extend the notion of assertion validity for partial predicates. But first we should admit that the presented definition will not be monotone under predicate extension. Indeed, consider informally the following assertion:

$$\{T\} \text{ while } T \text{ do skip } \{F\}.$$

This Floyd-Hoare triple will be true on all data, because the infinite loop is undefined on all data, and thus on all data the condition of validity for this assertion is satisfied. Now consider a triple $\{T\}$ skip $\{F\}$ that is false on all data. However, the mapping 'skip' is an extension of 'while T do skip'. Thus, monotonicity fails for a case when $p(d)=T$ and $f(d)$ is undefined. So, the value for this case should be changed.

To define a monotone interpretation of Floyd-Hoare triple for partial predicates we should change the question marks in Table 2 to Boolean values.

Table 2. Logical values of $\{p\}f\{q\}$ for partial predicates, where the question marks represent values that should be changed to proper Boolean values.

a) $f(d)$ is defined

$p(d) \setminus q(f(d))$	F	T	*Undefined*
F	T	T	?
T	F	T	?
undefined	?	?	?

b) $f(d)$ is undefined

$p(d)$	$\{p\}f\{q\}(d)$
F	T
T	?
undefined	?

To define such interpretation we adopt the following requirements:
- Monotonicity of a composition on all its arguments
- Maximal definiteness of the obtained predicates (we call this requirements as Kleene's principle)

We use techniques for non-deterministic semantics described in [7]. We treat the case when a predicate is 'undefined' as non-deterministic values T and F. Thus, we can use matrices from table 1 to evaluate a set of Boolean values for every case. The obtained results are presented in Table 3.

Table 3. Logical values of $\{p\}f\{q\}$ for partial predicates presented as sets of Boolean values.

a) $f(d)$ is defined

$p(d) \setminus q(f(d))$	$\{F\}$	$\{T\}$	$\{F,T\}$
$\{F\}$	$\{T\}$	$\{T\}$	$\{T\}$
$\{T\}$	$\{F\}$	$\{T\}$	$\{F,T\}$
$\{F,T\}$	$\{F,T\}$	$\{T\}$	$\{F,T\}$

b) $f(d)$ is undefined

$p(d)$	$\{p\}f\{q\}(d)$
$\{F\}$	$\{T\}$
$\{T\}$	$\{F,T\}$
$\{F,T\}$	$\{F,T\}$

Now, replacing non-deterministic results $\{F, T\}$ on undefined we get the final results (table 4).

Table 4. Logical values of $\{p\}f\{q\}$ for partial predicates.

a) $f(d)$ is defined

$p(d)\setminus q(f(d))$	F	T	*undefined*
F	T	T	T
T	F	T	*undefined*
undefined	*undefined*	T	*undefined*

b) $f(d)$ is undefined

$p(d)$	$\{p\}f\{q\}(d)$
F	T
T	*undefined*
undefined	*undefined*

The obtained matrices define an interpretation of $\{p\}f\{q\}$ for partial predicates. As was said earlier, we formalize such triples as a Floyd-Hoare composition $FH : Pr^{V,A} \times FPrg^{V,A} \times Pr^{V,A} \to Pr^{V,A}$ $(p, q \in Pr^{V,A}, f \in FPrg^{V,A}, d \in {}^{V}A)$:

$$FH(p,f,q)(d) = \begin{cases} T, & \text{if } q(f(d))\downarrow = T \text{ or } p(d)\downarrow = F, \\ F, & \text{if } p(d)\downarrow = T \text{ and } q(f(d))\downarrow = F, \\ & \text{undefined in other cases.} \end{cases}$$

2.4 Formal definition of program algebra compositions

In the previous subsection the formal definition of FH-composition was presented. In this subsection we give brief definitions of other compositions (see details in [3-5]).

Propositional compositions are defined as follows ($p,\ q \in Pr^{V,A},\ d \in {}^{V}A$):

$$(p \vee q)(d) = \begin{cases} T, \text{ if } p(d)\downarrow = T \text{ or } q(d)\downarrow = T, \\ F, \text{ if } p(d)\downarrow = F \text{ and } q(d)\downarrow = F, \\ \text{undefined in other cases.} \end{cases} \quad (\neg p)(d) = \begin{cases} T, \text{ if } p(d)\downarrow = F, \\ F, \text{if } p(d)\downarrow = T, \\ \text{undefined if } p(d)\uparrow. \end{cases}$$

Unary parametric composition of existential quantification ∃x with the parameter x∈V is defined by the following formula (p $\in \mathrm{Pr}^{V,A}$, d $\in {}^{V}$A):

$$(\exists x\ p)(d) = \begin{cases} T, & \text{if } b \in A \text{ exists}: p(d\nabla x \mapsto b)\downarrow = T, \\ F, & p(d\nabla x \mapsto a)\downarrow = F \text{ for each } a \in A, \\ & \text{undefined in other cases.} \end{cases}$$

Here $d\nabla x \mapsto a$ is a shorter form for $d\nabla[x \mapsto a]$.

Parametric n-ary superpositions with $\bar{x} = (x_1,...,x_n)$ as the parameter are defined by the following formulas ($f, g_1,\ldots, g_n \in Fn^{V,A},\ p \in Pr^{V,A}, d \in {}^{V}A$):

$$(S_F^{\bar{x}}(f, g_1,\ldots, g_n))(d) = f(d\nabla[x_1 \mapsto g_1(d),\ldots, x_n \mapsto g_n(d)]),$$

$$(S_P^{\bar{x}}(p, g_1,\ldots, g_n))(st) = f(st\nabla[x_1 \mapsto g_1(st),\ldots, x_n \mapsto g_n(st)]).$$

Null-ary parametric denomination composition with the parameter $x \in V$ is defined by the following formula ($d \in {}^{V}A$): $'x(d) = d(x)$.

Binary equality composition = is defined as follows ($f, g \in Fn^{V,A}, d \in {}^{V}A$):

$$(f{=}g)(d) = \begin{cases} T, \text{ if } f(d)\downarrow,\ g(d)\downarrow, \text{ and } f(d) = g(d), \\ F, \text{ if } f(d)\downarrow,\ g(d)\downarrow, \text{ and } f(d) \neq g(d), \\ \text{undefined in other cases.} \end{cases}$$

Identical program composition $id \in FPrg^{V,A}$ is the most simple: $id(d) = d$ ($d \in {}^{V}A$).

Assignment composition is defined as follows ($f \in Fn^{V,A}, d \in {}^{V}A$):

$$AS^x(f)(d) = d\nabla[x \mapsto f(d)].$$

Sequential execution is introduced in the ordinary way ($fs_1, fs_2 \in FPrg^{V,A}, d \in {}^{V}A$):

$$fs_1 \bullet fs_2(d) = fs_2(fs_1(d)).$$

Note, that we define • by commuting arguments of conventional functional composition: $fs_1 \bullet fs_2 = fs_2 \circ fs_1$.

Conditional composition depends on the value of the first function which is the condition itself ($p \in Pr^{V,A}, fs_1, fs_2 \in FPrg^{V,A}, d \in {}^{V}A$):

$$IF(p, fs_1, fs_2)(d) = \begin{cases} fs_1(d), \text{ if } p(d)\downarrow = T, \\ fs_2(d), \text{ if } p(d)\downarrow = F, \\ \text{undefined in other cases.} \end{cases}$$

Cycle is defined by the following formulas: $WH(p, fs)(d) = d_n$, where $d_0 = d$, $d_1 = fs(d_0)$, …, $d_n = fs(d_{n-1})$, moreover $p(d_0)\downarrow = T$, $p(d_1)\downarrow = T$, … , $p(d_{n-1})\downarrow = T$, ….. $p(d_n)\downarrow = F$ ($p \in Pr^{V,A}, fs \in FPrg^{V,A}, d \in {}^{V}A$).

It means that we have defined the following *quasiary program algebra*:

$QPA(V, A) = < Pr^{V,A}, Fn^{V,A}, FPrg^{V,A}; \vee, \neg, S_F^{\bar{v}}, S_P^{\bar{v}}, 'x, \exists x, =, id, AS^x, \bullet, IF, WH, FH>$.

This algebra is the main object of our investigation.

2.5 Formal definition of program algebra terms

Terms of the algebra $QPA(V, A)$ defined over sets of predicate symbols *Ps*, function symbols *Fs*, program symbols *Prs*, and variables V specify the syntax (the language) of the logic. We now give inductive definitions for terms $Tr(Ps, Fs, Prs, V)$, formulas $Fr(Ps, Fs, Prs, V)$, program texts $Pt(Ps, Fs, Prs, V)$, and Floyd-Hoare assertions $FHFr(Ps, Fs, Prs, V)$.

First we will define terms:

- if $f \in Fs$ then $f \in Tr(Ps, Fs, Prs, V)$
- if $v \in V$ then $'v \in Tr(Ps, Fs, Prs, V)$
- if $f \in Fs$, $t_1, \ldots, t_n \in Tr(Ps, Fs, Prs, V)$, and $v_1, \ldots, v_n \in V$ are distinct variables then $S_F^{\bar{v}}(f, t_1, \ldots, t_n) \in Tr(Ps, Fs, Prs, V)$

Then we will define program texts:

- $id \in Pt(Ps, Fs, Prs, V)$
- if $p \in Prs$ then $p \in Pt(Ps, Fs, Prs, V)$
- if $v \in V$ and $t \in Tr(Ps, Fs, Prs, V)$ then $AS^v(t) \in Pt(Ps, Fs, Prs, V)$
- if $p_1, p_2 \in Pt(Ps, Fs, Prs, V)$ then $p_1 \bullet p_2 \in Pt(Ps, Fs, Prs, V)$
- if $p_1, p_2 \in Pt(Ps, Fs, Prs, V)$ and $b \in Fr(Ps, Fs, Prs, V)$ then $IF(b, p_1, p_2) \in Pt(Ps, Fs, Prs, V)$
- if $p \in Pt(Ps, Fs, Prs, V)$ and $b \in Fr(Ps, Fs, Prs, V)$ then $WH(b, p) \in Pt(Ps, Fs, Prs, V)$

Finally, formulas and Floyd-Hoare triples are defined:

- if $p \in Ps$ then $p \in Fr(Ps, Fs, Prs, V)$
- if $\Phi \in Fr(Ps, Fs, Prs, V)$ then $\neg\Phi \in Fr(Ps, Fs, Prs, V)$
- if $t_1, t_2 \in Tr(Ps, Fs, Prs, V)$ then $t_1 = t_2 \in Fr(Ps, Fs, Prs, V)$
- if $\Phi \in Fr(Ps, Fs, Prs, V)$ and $v \in V$ then $\exists v\Phi \in Fr(Ps, Fs, Prs, V)$
- if $\Phi, \Psi \in Fr(Ps, Fs, Prs, V)$ then $\Phi \vee \Psi \in Fr(Ps, Fs, Prs, V)$;

– if $p \in Ps$, $t_1,\ldots,t_n \in Tr(Ps,Fs,Prs,V)$, and $v_1,\ldots,v_n \in V$ are distinct variables then $S_P^{\bar{v}}(p,t_1,\ldots,t_n) \in Fr(Ps,Fs,Prs,V)$
– if $f \in Pt(Ps,Fs,Prs,V)$ and $p,q \in Fr(Ps,Fs,Prs,V)$ then $\{p\}f\{q\} \in FHFr(Ps,Fs,Prs,V)$

After syntax and semantics have been defined, we need to specify the interpretation mappings, assuming that interpretation mappings for the predicate symbols $I_{Ps} : Ps \to Pr^{V,A}$, functional symbols $I_{Fs} : Fs \to Fn^{V,A}$, and program symbols $I_{Prs} : Prs \to FPrg^{V,A}$ are given. Let $J_{Fr} : Fr(Fs,Ps,Prs,V) \to Pr^{V,A}$ denote an interpretation mapping for formulas, $J_{Tr} : Tr(Fs,Ps,Prs,V) \to Fn^{V,A}$ denote an interpretation mapping for terms and $J_{Pt} : Pt(Fs,Ps,Prs,V) \to Prg^{V,A}$ denote an interpretation mapping for programs. They are all defined in a natural way, only the case with assertion needs special consideration:

$$J_{FHFr}(\{p\}f\{q\}) = FH(J_{Fr}(p),J_{Pt}(f),J_{Fr}(q)) .$$

An assertion is said to be *valid* (denoted $\models \{p\}f\{q\}$) if a corresponding predicate is *not refutable*.

3 Monotonicity and Continuity of the Floyd-Hoare Composition

In the previous section, a function-theoretic style of composition definitions was used. To prove properties of the FH-composition, it is more convenient to use a set-theoretic style of definition.

The following sets are called respectively *truth*, *false*, and *undefiniteness domains* of the predicate *p* over *D*:

$$p^T = \{d \mid p(d)\downarrow = T\},$$
$$p^F = \{d \mid p(d)\downarrow = F\},$$
$$p^\perp = \{d \mid p(d)\uparrow\}.$$

The following definitions introduce various images and preimages involved in Floyd-Hoare composition:

$$q^{-T,f} = f^{-1}[q^T],$$
$$q^{-F,f} = f^{-1}[q^F],$$
$$q^{-\perp,f} = f^{-1}[q^\perp],$$
$$p^{T,f} = f[p^T],$$
$$p^{F,f} = f[p^F],$$
$$p^{\perp,f} = f[p^\perp].$$

Using these notations we can define FH-composition by describing the truth and false domains of the predicate that is the value of the composition:

$$FH(p,f,q)^T = p^F \cup q^{-T,f},$$
$$FH(p,f,q)^F = p^T \cap q^{-F,f}.$$

Validity of formulas (predicates) is considered as irrefutability, that is $\models p \Leftrightarrow p^F = \varnothing$. From this follows that

$$\models FH(p,f,q) \Leftrightarrow p^T \cap q^{-F,f} = \varnothing.$$

Let us give a formal definition of the monotone composition.

Composition $C:(FPrg^{V,A})^n \times (Pr^{V,A})^k \times (Fn^{V,A})^m \to Pr^{V,A}$ is called *monotone* if the following condition holds for all arguments of *C*:

$$f_1 \subseteq g_1,\ldots,f_n \subseteq g_n, p_1 \subseteq q_1,\ldots,p_k \subseteq q_k, a_1 \subseteq b_1,\ldots,a_m \subseteq b_m \Rightarrow$$
$$C(f_1,\ldots,f_n,p_1,\ldots,p_k,a_1,\ldots,a_m) \subseteq C(g_1,\ldots,g_n,q_1,\ldots,q_k,b_1,\ldots,b_m).$$

Theorem 1. Floyd-Hoare composition is monotone on every argument.

Let us prove monotonicity on every argument separately, examining their truth and false domains. For truth domain we have:

$$p_1 \subseteq p_2 \Rightarrow {p_1}^T \subseteq p_2^T \Rightarrow {p_1}^T \cap q^{-F,f} \subseteq {p_2}^T \cap q^{-F,f} \Rightarrow$$
$$FH(p_1,f,q)^F \subseteq FH(p_2,f,q)^F.$$

Similar, for the false domain of the precondition we have:

$$p_1 \subseteq p_2 \Rightarrow {p_1}^F \subseteq p_2^F \Rightarrow {p_1}^F \cup q^{-T,f} \subseteq {p_2}^F \cup q^{-T,f} \Rightarrow$$
$$FH(p_1,f,q)^T \subseteq FH(p_2,f,q)^T.$$

Thus, $p_1 \subseteq p_2 \Rightarrow FH(p_1,f,q) \subseteq FH(p_2,f,q)$.

In the case of truth domain of postcondition the proof is similar:

$$q_1 \subseteq q_2 \Rightarrow {q_1}^T \subseteq q_2^T \Rightarrow q_1^{-T,f} \subseteq q_2^{-T,f} \Rightarrow p^F \cup q_1^{-T,f} \subseteq p^F \cup {q_2}^{-T,f} \Rightarrow$$
$$FH(p,f,q_1)^T \subseteq FH(p,f,q_2)^T.$$

The same for the false domain of postcondition:

$$q_1 \subseteq q_2 \Rightarrow {q_1}^F \subseteq q_2^F \Rightarrow q_1^{-F,f} \subseteq q_2^{-F,f} \Rightarrow p^T \cap q_1^{-F,f} \subseteq p^T \cap {q_2}^{-F,f} \Rightarrow$$
$$FH(p,f,q_1)^F \subseteq FH(p,f,q_2)^F.$$

Thus, $q_1 \subseteq q_2 \Rightarrow FH(p,f,q_1) \subseteq FH(p,f,q_2)$.

Let us show the monotonicity of the truth domains for the FP-composition:

$$f_1 \subseteq f_2 \Rightarrow q^{-T,f_1} \subseteq q^{-T,f_2} \Rightarrow p^F \cup q^{-T,f_1} \subseteq p^F \cup q^{-T,f_2}$$
$$\Rightarrow FH(p,f_1,q)^T \subseteq FH(p,f_2,q)^T.$$

Similar, for the false domains:

$$f_1 \subseteq f_2 \Rightarrow q^{-F,f_1} \subseteq q^{-F,f_2} \Rightarrow p^T \cap q^{-F,f_1} \subseteq p^T \cap q^{-F,f_2} \Rightarrow$$
$$FH(p,f_1,q)^F \subseteq FH(p,f_2,q)^F.$$

Also $f_1 \subseteq f_2 \Rightarrow FH(p,f_1,q) \subseteq FH(p,f_2,q)$.

Thus, it was shown that the composition is monotone on every component, what is needed to be proved.

For the constructed composition even stronger result is true, it is continuous. To show this, the following definitions are made and the notion of continuity is given (see, for example, [6]).

An infinite set of indexed functions (predicates) $\{f_0, f_1, \ldots\}, f_i \subseteq f_{i+1}, i \in \omega$ is called a *chain* of functions (predicates).

The supremum of the above-mentioned set of indexed functions (predicates) is called *limit* of the chain of functions (predicates), denoted as $\coprod_i f_i$.

The composition $C : (Prg^{V,A})^n \times (Pr^{V,A})^m \times (Fn^{V,A})^l \to Pr^{V,A}$ is called *continuous* on the first argument if for arbitrary chain $\{f_i \mid i \in \omega\}$ the following property holds: $C(\coprod_i f_i, g_2, \ldots, g_n, p_1, \ldots p_m, q_1, \ldots q_l) = \coprod_i C(f_i, g_2, \ldots, g_n, p_1, \ldots p_m, q_1, \ldots q_l)$.

Continuity on the other arguments is defined in a similar manner.

Theorem 2. Floyd-Hoare composition is continuous on every argument.

Though this result follows from the general consideration, we give here its direct proof. Let us show the continuity on the first argument. In the case of other arguments the proof will be similar.

Consider a chain of predicates $\{p_i \mid i \in \omega\}$. Since Floyd-Hoare composition is monotone, $\{FH(p_i, f, q) \mid i \in \omega\}$ will also be a chain. We need to show that $FH(\coprod_i p_i, f, q) = \coprod_i FH(p_i, f, q)$.

For the arbitrary data d, there are two different possibilities – $\coprod_i p_i(d) \uparrow$ and $\coprod_i p_i(d) \downarrow$. In the first case none of the elements of the chain in defined on d. Thus $\forall j \in \omega, FH(\coprod_i p_i, f, q)(d) = FH(p_j, f, q)(d)$, therefore needed equality is obvious. If the limit is defined on these data, an element of the chain that is also defined on this data could be found. Otherwise the limit would have been undefined on those data, what is guaranteed by the inclusion relation on the elements of the chain. Let the limit be the element with index *k*. Then

$$FH(\coprod_i p_i, f, q)(d) = FH(p_k, f, q)(d) \text{ and}$$

$$FH(p_k, f, q)(d) = \coprod_i FH(p_i, f, q)(d),$$

since $\forall i > k, p_i(d) = p_k(d)$ from the definition of the chain.

The following equality is obtained: $FH(\coprod_i p_i, f, q)(d) = \coprod_i FH(p_i, f, q)(d)$. Since the data was chosen arbitrary, we get $FH(\coprod_i p_i, f, q) = \coprod_i FH(p_i, f, q)$, what was needed to be proved.

The proof for the other arguments (a program and a postcondition) is similar. Thus, it is proven that the monotone Floyd-Hoare composition is also continuous on every argument.

4 Soundness of Inference Rules System in Floyd-Hoare Algebras

In this section we adopt the same convention as earlier that we do not distinguish between syntactic and semantic notation for formulas. We also assume that the algebra $QPA(V, A)$ is fixed and interpretation mappings are also fixed.

Since a result of the Floyd-Hoare composition can be undefined on some data, classical inference rules can be unsound. This informally means that with true preconditions they could give false postconditions. This happens because predicates can be partial and compositions are defined in a way that differs from the classical Floyd-Hoare composition to be monotone. Let us examine the following system of inference rules to find out what conditions are required for rules to be sound:

$$\{S^{[x]}(p,f)\}AS^{x}(f)\{p\} \ - Ax\_AS$$

$$\{p\}id\{p\} \ - Ax\_ID$$

$$\frac{\{p\}f\{q\},\{q\}g\{r\}}{\{p\}f \bullet g\{r\}} \ - Ax\_SEQ$$

$$\frac{\{b \wedge p\}f\{q\},\{\neg b \wedge p\}g\{q\}}{\{p\}IF(b,f,g)\{q\}} \ - Ax\_IF$$

$$\frac{\{b \wedge p\}f\{p\}}{\{p\}WH(b,f)\{\neg b \wedge p\}} \ - Ax\_WH$$

$$\frac{\{p'\}f\{q'\}}{\{p\}f\{q\}} \ - Ax\_CONS$$

Note that we do not include additional conditions for the consequence rule, because in different classes of algebras we will have different conditions.

An assertion $\{p\}f\{q\}$ is said to be *derived* if there exists its derivation tree with rules of the type *Ax_AS*, *Ax_ID* on its leaves. Derivability is denoted as $\vdash \{p\}f\{q\}$.

Let us show that for the rules *Ax_SEQ*, *Ax_WH*, and *Ax_CONS* without additional conditions we can give such an example of the application of the inference rule that will have true preconditions and false postconditions.

Consider *Ax_SEQ* with violation of the condition $p^{T,f} \subseteq q^{T}$.

If this condition fails then $\exists p,q,f,d : p(d) = T, q(f(d))\uparrow, \models \{p\}f\{q\}$. In this case we will take such r and g that $\models \{q\}g\{r\}$ and $g(f(d))\downarrow, r(g(f(d))) = F$. This is possible if we define them in the following way:

$$g = id, \ r(x) = \begin{cases} T, x \neq f(d), \\ F, x = f(d). \end{cases}$$

Then $\models \{p\}f \bullet g\{r\}$ does not hold, while $p(d) = T$ and $r(f \bullet g(d)) = F$, what is equal to $\{d\} \subseteq p^{T} \cap r^{-F,f \bullet g}$.

Consider *Ax_WH* with violation of the condition $(b \wedge p)^{T,f} \subseteq p^{T}$.

We will construct such b, f, and p that the following properties hold:

$$\models \{b \wedge p\} f \{p\}, (b \wedge p)^{T,f} \not\subset p^T,$$
$$\not\models \{p\} WH(b,f) \{\neg b \wedge p\}.$$

Let $d_1 \neq d_2 \neq d_3$. Then b, f, and p are defined in the following manner:

$$b(x) = \begin{cases} T, x \neq d_3, \\ F, x = d_3. \end{cases}$$

$$f(x) = \begin{cases} x, x \neq d_1, d_2, \\ d_2, x = d_1, \\ d_3, x = d_2. \end{cases}$$

$$p(x) = \begin{cases} T, x \neq d_2, d_3, \\ \perp, x = d_2, \\ F, x = d_3. \end{cases}$$

It is not hard to check that the above-mentioned properties are not satisfied:

$$d_2 \in (b \wedge p)^{T,f}, d_2 \notin p^T,$$
$$d_1 \in p^T, d_3 = WH(b,f)(d_1),$$
$$d_3 \in (\neg b \wedge p)^F \Rightarrow d_1 \in (\neg b \wedge p)^{-F, WH(b,f)},$$
$$d_1 \in (b \wedge p)^T, d_2 \in p^{\perp}, d_3 \in p^F.$$

Thus, $\models \{b \wedge p\} f \{p\}$ because for other data p is true.

That proves that the additional condition is necessary because in other cases the rule is not sound while used on such examples.

The case with the rule *Ax_CONS* is similar to the previous one with the conditions $p^T \subseteq p'^T, q^F \subseteq q'^F$.

So, it was shown that additional conditions are not redundant. Let us show that if additional conditions hold then the rules are sound.

Theorem 3. Inference rules are sound with additional conditions. In other words:

$$\models \{S^{[x]}(p,f)\} AS^x(f) \{p\},$$
$$\models \{p\} id \{p\},$$
$$\models \{p\} f \{q\} \wedge \models \{q\} g \{r\} \wedge p^{T,f} \subseteq q^T \Rightarrow \models \{p\} f \bullet g \{r\},$$
$$\models \{b \wedge p\} f \{q\} \wedge \models \{\neg b \wedge p\} g \{q\} \Rightarrow \models \{p\} IF(b,f,g) \{q\},$$
$$\models \{b \wedge p\} f \{p\} \wedge (b \wedge p)^{T,f} \subseteq p^T \Rightarrow \models \{p\} WH(b,f) \{\neg b \wedge p\},$$
$$\models \{p'\} f \{q'\} \wedge p^T \subseteq p'^T \wedge q^F \subseteq q'^F \Rightarrow \models \{p\} f \{q\}.$$

Let us prove this for each rule.

For $\models \{S^{[x]}(p,f)\} AS^x(f) \{p\}$ to hold it is needed that the following condition holds: $FH(S^{[x]}(p,f), AS^x(f), p)^F = (S^{[x]}(p,f))^T \cap p^{-F, AS^x(f)} = \varnothing$.

Assume that it is false and the intersection is not empty. Let some data d belongs to the intersection.

If $d \in (S^{[x]}(p,f))^T$ then $p(d\nabla[x \mapsto f(d)]) = T$.

Let $d \in p^{-F,AS^x(f)}$ then $p(AS^x(f)(d)) = p(d\nabla[x \mapsto f(d)]) = F$, what is impossible, thus, the assumption is incorrect and $(S^{[x]}(p,f))^T \cap p^{-F,AS^x(f)} = \varnothing$, similar, $(S^{[x]}(p,f))^{T,AS^x(f)} \cap p^{\perp} = \varnothing$, what means $\models \{S^{[x]}(p,f)\}AS^x(f)\{p\}$.

$\models \{p\}id\{p\}$ follows from the definition.

Let us prove $\models \{p\}f\{q\} \wedge \models \{q\}g\{r\} \wedge p^{T,f} \subseteq q^T \Rightarrow \models \{p\}f \bullet g\{r\}$.

We have $\models \{p\}f\{q\}, \models \{q\}g\{r\}$ that means $p^T \cap q^{-F,f} = \varnothing; q^T \cap r^{-F,g} = \varnothing$. We need to show that $p^T \cap r^{-F,f \bullet g} = \varnothing$.

Let it be false and $\exists d : d \in p^T \wedge d \in r^{-F,f \bullet g}$. This means that $p(d) = T \wedge r(f \bullet g(d)) = F$.

But using the additional condition we have $p^{T,f} \subseteq q^T$, thus $q(f(d)) = T$. That means $f(d) \in q^T$, then $f(d) \notin r^{-F,g}$. This contradicts the fact that $r(f \bullet g(d)) = F \Rightarrow f \bullet g(d) \in r^F \Rightarrow f(d) \in r^{-F,g}$.

We have the contradiction, which means that the assumption is wrong and $p^T \cap r^{-F,f \bullet g} = \varnothing$. Then $\models \{p\}f \bullet g\{r\}$.

Let us prove $\models \{b \wedge p\}f\{q\} \wedge \models \{\neg b \wedge p\}g\{q\} \Rightarrow \models \{p\}IF(b,f,g)\{q\}$.

We have $\models \{b \wedge p\}f\{q\}$, $\models \{\neg b \wedge p\}g\{q\}$, which means:
$(b \wedge p)^T \cap q^{-F,f} = \varnothing; (\neg b \wedge p)^T \cap q^{-F,g} = \varnothing$.

We need to show that $p^T \cap q^{-F,IF(b,f,g)} = \varnothing$.

Let $\exists d : d \in p^T \wedge d \in q^{-F,IF(b,f,g)}$. Then $p(d) = T, q(IF(b,f,g)(d)) = F$.

Let us examine different cases of $b(d)$:

$b(d)\uparrow$ is impossible, because then $IF(b,f,g)(d)\uparrow$ leads to a contradiction with assumptions about existence of such d.

$b(d) = T \Rightarrow IF(b,f,g)(d) = f(d) \wedge d \in (b \wedge p)^T \wedge IF(b,f,g)(d) \in (b \wedge p)^{T,f}$.

With properties of the upper part of the inference rule we have:

$(b \wedge p)^T \cap q^{-F,f} = \varnothing \wedge IF(b,f,g)(d) = f(d) \wedge d \in (b \wedge p)^T \Rightarrow d \notin q^{-F,(IF(b,f,g)}$.

A case with $b(d) = F$ is similar to the case where $b(d) = T$.

$(\neg b \wedge p)^T \cap q^{-F,g} = \varnothing \wedge IF(b,f,g)(d) = g(d) \wedge d \in (\neg b \wedge p)^T \Rightarrow d \notin q^{-F,(IF(b,f,g)}$.

Thus $d \notin q^{-F,(IF(b,f,g)}$ in any case if d is defined which is guaranteed by the assumption. That leads us to the contradiction, so, $p^T \cap q^{-F,IF(b,f,g)} = \varnothing$.

Thus, we have $\models \{p\}IF(b,f,g)\{q\}$.

Let us prove $\models \{b \wedge p\}f\{p\} \wedge (b \wedge p)^{T,f} \subseteq p^T \Rightarrow \models \{p\}WH(b,f)\{\neg b \wedge p\}$.

We have $\models \{b \wedge p\}f\{p\}$, that means: $(b \wedge p)^T \cap p^{-F,f} = \varnothing$.

We need to show that the following condition holds: $p^T \cap (\neg b \wedge p)^{-F,WH(b,f)} = \varnothing$.

Let $\exists d : d \in p^T \wedge d \in (\neg b \wedge p)^{-F,WH(b,f)}$. Then $\exists d_n : d_n = WH(b,f)(d)$, and by the definition of the composition we have $\exists d_1, d_2, \ldots d_n : \quad d = d_1 \quad \wedge d_{i+1} = f(d_i)$, $i = \overline{1,n-1} \wedge b(d_j) = T$, $j = \overline{1,n-1} \wedge p(d_1) = T \wedge b(d_n) = F$ and $(\neg b \wedge p)(d_n) = F$.

Thus, $(b \wedge p)(d_1) = T$. By $(b \wedge p)^{T,f} \subseteq p^T$ we obtain that $p(d_2) = p(f(d_1)) = T$.

Using the induction over a number of loop execution we obtain that $p(d_n) = T$. That means $(\neg b \wedge p)(d_n) = T$. Thus, we obtained contradiction, so, $p^T \cap (\neg b \wedge p)^{-F,WH(b,f)} = \varnothing$.

Let us prove $\models \{p'\} f \{q'\} \wedge p^T \subseteq p'^T \wedge q^F \subseteq q'^F \Rightarrow \models \{p\} f \{q\}$.

We have $\models \{p'\} f \{q'\}$, what means: $p'^T \cap q'^{-F,f} = \varnothing$.

We need to show that $p^T \cap q^{-F,f} = \varnothing$.

Let this condition be false and $\exists d : d \in p^T \wedge d \in q^{-F,f}$. This means that $p(d) = T \wedge q(f(d)) = F$.

By the condition that $p^T \subseteq p'^T$ we obtain $p'(d) = T$.

But $p'^T \cap q'^{-F,f} = \varnothing$, thus, $d \notin q'^{-F,f}$, while $d \in q^{-F,f}$, then $f(d)\downarrow$, and we have $q'(f(d)) \neq F$.

We have a contradiction which means that the assumption does not hold, so, $p^T \cap q^{-F,f} = \varnothing$.

Both conditions are proved, then $\models \{p\} f \{q\}$.

Thus, all rules are inspected and theorem is proved.

Also the condition for the rule *Ax_SEQ* can be substituted by one of the following: $p^{T,f} \cap q^{\perp} = \varnothing$, $q^{\perp,g} \cap r^F = \varnothing$ or $q^{F,g} \supseteq r^F$, but none of them is a sufficient condition, because $(\models \{p\} f \{q\} \wedge \models \{q\} g \{r\} \Rightarrow \models \{p\} f \bullet g \{r\}) \Rightarrow p^{T,f} \subseteq q^T$ doesn't hold.

Similar for the rule *Ax_WH*, the condition could be given in the one of the following manner: $(b \wedge p)^{T,f} \cap p^{\perp} = \varnothing$, $(b \wedge p)^{\perp,f} \cap p^F = \varnothing$ or $(b \wedge p)^{F,f} \supseteq p^F$, and they are also insufficient.

The conditions for the rule *Ax_CONS* also are not sufficient. To prove that we need only to show an example when the condition does not hold but the rule does.

But in some cases we can avoid adding the conditions implicitly to the rules.

Theorem 4. For all assertions $\{p\} f \{q\}$ that were inferred using rules of the inference system except *Ax_CONS* the following properties hold:

$$p^{T,f} \subseteq q^T,$$
$$p^{T,f} \cap q^{\perp} = \varnothing,$$
$$p^{\perp,f} \cap q^F = \varnothing,$$
$$p^{F,f} \supseteq q^F.$$

Let us prove the first property by induction. For the fourth property the case is similar and second and third properties are consequences of the first and the fourth respectively.

Induction base: for *Ax_ID* and *Ax_AS* proof is obvious.

Induction step. For *Ax_SEQ* we have:

$$p^{T,f} \subseteq q^T \wedge q^{T,g} \subseteq r^T \Rightarrow p^{T,f \bullet g} \subseteq r^T .$$

The proof of this fact is obvious.

For *Ax_IF* we need to prove:

$$(b \wedge p)^{T,f} \subseteq q^T \wedge (\neg b \wedge p)^{T,g} \subseteq (q)^T \Rightarrow p^{T,IF(b,f,g)} \subseteq q^T .$$

Consider $d \in p^{T,IF(b,f,g)}$, then $\exists x : (IF(b,f,g)(x) = d) \wedge p(x)$, that leads to two cases:

– $b(x) = T$, then $f(x) \in q^T$, moreover $d = IF(b,f,g)(x) = f(x)$, thus, $d \in q^T$;

– $b(x) = F$, then $g(x) \in q^T$, moreover $d = IF(b,f,g)(x) = g(x)$, thus, $d \in q^T$.

For *Ax_WH* we need to prove: $(b \wedge p)^{T,f} \subseteq p^T \Rightarrow p^{T,WH(b,f)} \subseteq (\neg b \wedge p)^T$.

Let $d \in p^{T,WH(b,f)}$, then $\exists x : (WH(b,f)(x) = d) \wedge p(x)$, we need to prove that $(\neg b \wedge p)(d) = T$. Let us examine all data that are obtained during the calculation of $WH(b,f)(x)$: $x = x_0$; $x_1 = f(x_0)$ ... $d = x_n$, $b(x_0) = b(x_1) = \ldots b(x_{n-1}) = T$, $b(x_n) = F$, thus from $(b \wedge p)^{T,f} \subseteq p^T$ we have, that $p(d) = p(x_n) = T$, this together with $b(x_n) = F$ gives $(\neg b \wedge p)(d) = T$, what was needed to prove.

The theorem is proved.

Theorem 3 and Theorem 4 together give us the fact that if we declare *Ax_CONS* in such a way that it retains the properties of the theorem 4, then inference rules system will be sound without addition of new conditions, which will be guaranteed by Theorem 4.

But in this case system would not be complete. Let us give an example.

Let q be an arbitrary predicate that has nonempty truth, false, and undefiniteness domains, and p be such predicate that $p^T = q^T \cup q^{\perp}$. Then $\models \{p\} id \{q\}$, but $\neg(p^{T,id} \subseteq q^T)$, when the inference rules system was constructed for the following property to hold: $\vdash \{p\} f \{q\} \Rightarrow p^{T,f} \subseteq q^T$.

5 Conclusions

In this paper special program algebras of partial quasiary mappings have been described. Such algebras form a semantic base for a modified Floyd-Hoare logic. In this case assertions have been presented by a special composition called Floyd-Hoare composition. Monotonicity and continuity of this composition have been proved. The language of the modified Floyd-Hoare logic has been described. Further, the inference rules for such a logic have been studied and their soundness conditions have been specified. The logic constructed can be used for program verification.

The major directions of further investigation are the question of completeness of the system of inference rules, invariants for rules, and types for variables and functions. Also the authors plan to construct a prototype of a program system in the style of [8, 9] oriented on the constructed logics.

References

1. Floyd, R. W.: Assigning Meanings to Programs. In: Proc. American Mathematical Society Symposia on Applied Mathematics, vol. 19, pp. 19–31 (1967)
2. Hoare, C. A. R.: An Axiomatic Basis for Computer Programming. Comm. ACM, 12, 576–580, 583 (1969)
3. Nikitchenko, M. S., Shkilniak, S. S.: Mathematical Logic and Theory of Algorithms. Publishing house of Taras Shevchenko National University of Kyiv, Kyiv (2008) (in Ukrainian)
4. Nikitchenko, M., Tymofieiev, V.: Satisfiability and Validity Problems in Many-Sorted Composition-Nominative Pure Predicate Logics. In: V. Ermolayev et al. (eds.): ICTERI 2012, CCIS 347, pp. 89–110. Springer Verlag, Berlin Heidelberg (2013)
5. Nikitchenko, M. S., Tymofieiev, V. G.: Satisfiability in Composition-Nominative Logics. Central European Journal of Computer Science, 2(3), 194–213 (2012)
6. Nielson. H.R., Nielson, F.: Semantics with Applications: A Formal Introduction. John Wiley & Sons Inc. (1992)
7. Avron, A., Zamanskym A.: Non-Deterministic Semantics for Logical Systems. Handbook of Philosophical Logic, vol. 16, pp. 227–304 (2011)
8. Schreiner, W.: Computer-Assisted Program Reasoning Based on a Relational Semantics of Programs. In: P. Quaresma and R.-J. Back (eds.) Proc 1st Workshop on CTP Components for Educational Software (THedu'11), July 31 2011, Wrocław, Poland, No 79 of Electronic Proceedings in Theoretical Computer Science (EPTCS), ISSN: 2075-2180, pp. 124–142 (2012)
9. Schreiner, W.: A Program Calculus Technical Report. Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, http://www.risc.uni-linz.ac.at/people/schreine/papers/ProgramCalculus2008.pdf (2008)