PROPERTIES OF INFERENCE SYSTEMS FOR FLOYD-HOARE LOGIC WITH PARTIAL PREDICATES

Mykola NIKITCHENKO, Andrii KRYVOLAP
*Department of Theory and Technology of Programming, Taras Shevchenko National University of Kyiv,
64/13, Volodymyrska Street, Kyiv, Ukraine, 01601, tel. +380 442 393 333,
e-mail: nikitchenko@unicyb.kiev.ua, krivolapa@gmail.com

ABSTRACT

The main object of research in this paper is extension of Floyd-Hoare logic on partial pre- and postconditions. Composition-nominative approach is used to define this extension. According to this approach semantics of extended logic is represented by algebras of partial quasiary mappings. Operations of these algebras are called compositions. Semantics of Floyd-Hoare triples (assertions) is formalized with the help of a special monotone composition. Preimage predicate transformer composition – a counterpart of the weakest precondition – is also defined. Sound modifications of inference systems with added constraints are presented. Properties of extensional and intensional completeness of such inference systems are studied.

Keywords: *Floyd-Hoare logic, precondition and postcondition, partial predicate, inference system, soundness and completeness, composition-nominative approach, predicate transformer.*

1. INTRODUCTION

Floyd-Hoare logic [1, 2] is one of the formal systems used for reasoning about program correctness. The basic notion of this logic is Floyd-Hoare triple (assertion). It consists of a precondition, a program, and a postcondition. The idea of such triples stems from the following requirement for programs: when the input meets the precondition the program should return the output that meets the postcondition (if the program terminates). Reasoning about program properties in terms of Floyd-Hoare triples is convenient and natural. Predicates in classical Floyd-Hoare logic are assumed to be total. In general case predicates that are used to describe the needed properties of programs can be partial. In this case additional techniques are required to transform partial predicates to total. Such transformations complicate the use of Floyd-Hoare logic. This explains the demand for extension of Floyd-Hoare logic on partial predicates. For such extensions the results obtained for total predicates should remain sound.

Special Floyd-Hoare composition is used to represent semantics of assertions. This composition has three arguments: a predicate as a precondition, a program function, and a predicate as a postcondition. The output is a predicate. The classical definition of Floyd-Hoare triple validity leads to Floyd-Hoare composition that is not monotone [3]. Monotonicity is one of the key properties for reasoning about programs. In particular, monotonicity is important for acyclic (loop-free) programs used for approximations of programs with cycles. This explains the need of a new definition of Floyd-Hoare composition for the extension of Floyd-Hoare logic on partial predicates. It should not only be monotone, but also converge to the classical definition if predicates are total. Such definition was presented in [3].

One of the possible ways to apply Floyd-Hoare logic in program verification is to develop an inference system depending on the language under consideration. It was shown [3] that classical inference system for the simple imperative language WHILE [4] is neither sound nor complete if extended on partial predicates as it is. But soundness and completeness are the main properties of an inference system that justify its applicability. With an inference system that is unsound there are no guarantees that derived assertions are valid. In the case of incomplete inference system, there is no derivation for some of the valid assertions. In classical Floyd-Hoare logic with total predicates the abovementioned inference system for the language WHILE is sound and complete (in the extensional approach) [4]. Completeness in the extensional approach means that pre- and postconditions may be arbitrary predicates. In the intensional approach pre- and postconditions additionally should be presented by formulas of a given language. The problem with soundness and completeness arises when partial predicates are taken into consideration. In order to adopt inference system for the language WHILE in classical Floyd-Hoare logic to the extension of the logic on partial predicates maintaining soundness of the system, additional constraints on inference rules should be introduced that correspond to the new definition of validity of Floyd-Hoare assertions. When a sound inference system is introduced, its extensional completeness is the subject of further investigation. For this purpose a special composition is additionally defined. Inspired by the weakest predicate transformer, introduced by Dijkstra, this composition is called preimage predicate transformer composition.

To give the answer to the question, whether introduced systems with added constraints are intensionally complete, first only acyclic programs are considered. Acyclic programs are treated as approximations of the programs with cycles. For acyclic programs, predicates obtained with the preimage predicate transformer composition can be transformed to formulas of first-order quasiary predicate logic, thus the validity problem in extended Floyd-Hoare logic can be reduced first to the validity problem in first-order quasiary predicate logic, then to the validity problem in classical predicate logic [6].

In this paper we continue our research of Floyd-Hoare logics with partial predicates initiated in [3], concentrating on such properties of inference systems as their extensional and intensional completeness. The notions not defined here

are understood in the sense of [3].

2. SEMANTICS OF FLOYD-HOARE LOGIC WITH PARTIAL PREDICATES

We will use semantic-syntactic approach to define an extended logic. It means first that algebras of partial mapping will be specified to present the semantics of the extended Floyd-Hoare logic. Then the syntax or the language of the extended logic will result from the definition of semantics and will be simply the set of terms of introduced algebras. That is why for simplicity's sake in this paper we will use the same notation for predicates and terms that represent predicates.

Composition-nominative approach [5] is used for defining semantics. All data are treated as nominative sets (nominative data in general case). *Nominative sets* are defined as partial mappings from a set of names (variables) to a set of basic values. Such mappings do not have fixed arity and are called *quasiary*. Nominative sets can be also treated as states of program variables. More complex case of hierarchical nominative data is not considered in this paper and is a subject of separate investigation. Compositionality means that complex functions and predicates are built from simpler ones using compositions. Compositions are the operations of respective algebras used in defining semantics. Also all mappings are partial (can be undefined on some data).

We start with definitions of the nominative sets, quasiary predicates and functions.

The arrows $\xrightarrow{p}$ and $\xrightarrow{t}$ specify the sets of partial and total mappings respectively. Also for an arbitrary partial mapping $f : D \xrightarrow{p} D'$:

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

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

$${}^V A = V \xrightarrow{p} A$$

Set-like notation for nominative sets is more convenient in some cases. We will use the following notation: $[v_i \mapsto a_i \mid i \in I]$ to describe a nominative set where variables v_i have values a_i respectively. Thus, $v_i \mapsto a_i \in_n d$ denotes that $d(v_i)\downarrow = a_i$ or in other words that value of the variable v_i in nominative set d is a_i $(i \in I)$.

One of the main operations is *overriding* operation. It is a binary total operation that joins two nominative sets taking into account names of the variables, and is defined in the following way:

$$d_1 \nabla d_2 = [v \mapsto a \mid v \mapsto a \in_n d_2 \vee (v \mapsto a \in_n d_1 \wedge \\ \wedge \neg \exists a' (v \mapsto a' \in_n d_2))] \quad (1)$$

Informally this means that all name-value pairs from d_2 and those pairs from d_1 whose names are not defined in d_2 are present in the resulting nominative set.

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*. They represent different conditions in programs.

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 ordinary quasiary functions*. They represent different expressions in programs. The term 'ordinary' is used to distinguish ordinary functions from program functions (bi-quasiary functions) that represent programs. This term will usually be omitted.

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. They represent semantics of programs.

Algebras with three presented sets (partial quasiary predicates, partial ordinary quasiary functions, and partial bi-quasiary functions) as algebra carriers (sorts) can be used to define semantics of the logics. We distinguish three logics of different levels of abstraction:

- pure predicate logics based on algebras with one sort $Pr^{V,A}$ of quasiary predicates;
- quasiary predicate-function logics based on algebras with two sorts: $Pr^{V,A}$ of quasiary predicates and $Fn^{V,A}$ of quasiary functions;
- quasiary program logics based on algebras with three sorts: $Pr^{V,A}$ of quasiary predicates; $Fn^{V,A}$ of quasiary functions, and $FPrg^{V,A}$ of bi-quasiary functions.

At the level of pure predicate logics only predicates are taken into consideration. Basic compositions represent basic logic connectives such as disjunction $\vee : Pr^{V,A} \times Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ and negation $\neg : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$. Also parametric quantification $\exists x : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ and renomination $R^{\bar{v}}_{\bar{x}} : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ compositions [5,6] should be included to the basic compositions (here $\bar{v}$ stands for $v_1, \ldots, v_n$ and $\bar{x}$ for $x_1, \ldots, x_n$). Renomination is a specific new composition for quasiary predicates. Informally, while evaluating $R^{\bar{v}}_{\bar{x}}(p)(d)$ we construct a new nominative set changing in d values of names from $\bar{v}$ with values of corresponding names from $\bar{x}$; then p is evaluated on the obtained nominative set.

In quasiary predicate-function logics we add ordinary quasiary functions to the scope. Basic compositions of the pure predicate logics are extended with parametric compositions of superposition for functions $S^{\bar{x}}_F : (Fn^{V,A})^{n+1} \xrightarrow{t} Fn^{V,A}$ and predicates

$S_P^{\bar{x}} : Pr^{V,A} \times (Fn^{V,A})^n \xrightarrow{t} Pr^{V,A}$ [5,6]. Another basic composition that has to be added is null-ary parametric composition of denomination $'x : Fn^{V,A}$. Renomination composition can be given as a combination of superposition and denomination compositions, thus $R_{\bar{x}}^{\bar{y}}(p) = S_P^{\bar{y}}(p,'x_1,\ldots,'x_n)$. So, renomination compositions can be omitted.

When bi-quasiary functions are taken into consideration on the level of program logics there are many possible ways to define compositions that provide means to construct complex programs from simpler ones. We have chosen the following compositions of the language WHILE to include them as basic to the logics of program level: the parametric assignment composition $AS^x : Fn^{V,A} \xrightarrow{t} FPrg^{V,A}$, which corresponds to assignment operator := ; identity composition (function) $id : FPrg^{V,A}$, which corresponds to the *skip* operator of WHILE language; composition of sequential execution $\bullet : FPrg^{V,A} \times FPrg^{V,A} \xrightarrow{t} FPrg^{V,A}$; conditional composition $IF : Pr^{V,A} \times FPrg^{V,A} \times FPrg^{V,A} \xrightarrow{t} FPrg^{V,A}$, which corresponds to the operator *if_then_else*; cyclic composition $WH : Pr^{V,A} \times FPrg^{V,A} \xrightarrow{t} FPrg^{V,A}$, which corresponds to the operator *while_do*.

We also need compositions that could provide possibility to construct predicates describing some properties of programs. The main composition of this kind is Floyd-Hoare composition $FH : Pr^{V,A} \times FPrg^{V,A} \times Pr^{V,A} \xrightarrow{t} Pr^{V,A}$. It takes precondition, postcondition, and program as inputs and yields a predicate that represents respective Floyd-Hoare assertion.

Now we will give formal definitions of the compositions of these algebras. In the following definitions $d \in {}^{V}A$, $f, g_1, \ldots, g_n \in Fn^{V,A}$, $p, q, r \in Pr^{V,A}$, $\bar{x} = (x_1, \ldots, x_n) \in V^n$, $x \in V$, $pr_1, pr_2, pr \in FPrg^{V,A}$.

$$(p \vee q)(d) = \begin{cases} T, \text{ if } p(d) \downarrow = T \text{ or } q(d) \downarrow = T, \\ F, \text{ if } p(d) \downarrow = F \text{ and } q(d) \downarrow = F, \\ \text{undefined in other cases.} \end{cases} \quad (2)$$

$$(\neg p)(d) = \begin{cases} T, \text{ if } p(d) \downarrow = F, \\ F, \text{ if } p(d) \downarrow = T, \\ \text{undefined in other cases.} \end{cases} \quad (3)$$

$$(\exists x p)(d) = \begin{cases} T, \text{ if } p(d\nabla x \mapsto a) \downarrow = T \text{ for some } a \in A, \\ F, \text{ if } p(d\nabla x \mapsto a) \downarrow = F \text{ for each } a \in A, \\ \text{undefined in other cases.} \end{cases} \quad (4)$$

$$S_P^{\bar{x}}(p, g_1, \ldots, g_n)(d) = \\ = p(d\nabla[x_1 \mapsto g_1(d), \ldots, x_n \mapsto g_n(d)]) \quad (5)$$

$$S_F^{\bar{x}}(f, g_1, \ldots, g_n)(d) = \\ = f(d\nabla[x_1 \mapsto g_1(d), \ldots, x_n \mapsto g_n(d)]) \quad (6)$$

$$'x(d) = d(x) \quad (7)$$

$$AS^x(f)(d) = d\nabla[x \mapsto f(d)] \quad (8)$$

$$id(d) = d \quad (9)$$

$$pr_1 \bullet pr_2(d) = pr_2(pr_1(d)) \quad (10)$$

$$IF(r, pr_1, pr_2)(d) = \begin{cases} pr_1(d), \text{ if } r(d) \downarrow = T \text{ and } pr_1(d) \downarrow, \\ pr_2(d), \text{ if } r(d) \downarrow = F \text{ and } pr_2(d) \downarrow, \\ \text{undefined in other cases.} \end{cases} \quad (11)$$

$$WH(r, pr)(d) = d_n, \text{ if } r(d) \downarrow = T, f(d) \downarrow = d_1, \\ r(d_1) \downarrow = T, f(d_1) \downarrow = d_2, \ldots, f(d_{n-1}) \downarrow = d_n, r(d_n) \downarrow = F \quad (12)$$

$$FH(p, pr, q)(d) = \begin{cases} T, \text{ if } p(d) \downarrow = F \text{ or } q(pr(d)) \downarrow = T, \\ F, \text{ if } p(d) \downarrow = T \text{ and } q(pr(d)) \downarrow = F, \\ \text{undefined in other cases.} \end{cases} \quad (13)$$

Floyd-Hoare composition was proved to be not only monotone but also continuous [3]. Formal definition of monotonicity of Floyd-Hoare composition can be given as follows:

$$p \subseteq p', q \subseteq q', pr \subseteq pr' \Rightarrow \\ \Rightarrow FH(p, pr, q) \subseteq FH(p', pr', q') \quad (14)$$

Inclusion relation $\subseteq$ is understood here as inclusion of the graphs of the functions or predicates. Monotonicity of other compositions can be defined in the same manner. Definition of continuity is omitted in this paper. Defined program algebras form semantic basis of the Floyd-Hoare logic extension on partial predicates.

Thus, the following basic algebras (for different A) represent semantics of basic program logics:

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

Such basic program logics are used to formulate program assertions but they are expressively weak to specify proofs of such properties. Therefore we should define more expressive algebras, and, consequently, more expressive languages. These algebras will be used to study completeness of the constructed logics. The expressive power of such algebras is specified by their compositions.

First, we introduce a new composition which is inspired by the weakest precondition introduced by Dijkstra [7]. The main idea of the weakest precondition predicate transformer was to define a predicate that would be a valid precondition, when two other arguments of Floyd-Hoare assertion are given. It is obvious that there are different preconditions that stem from this idea, but not all of them are

of a practical importance. That is why only the weakest precondition is taken into consideration. Usually "weakest" is specified in terms of logical consequence. But at the same time for partial predicates different definitions of logical consequence can be applied to define the notion of weakest precondition. Therefore we will not put much emphasis on these notions but specify the required composition as *preimage predicate transformer composition* (*PC*) that makes the assertions of the form $\{PC(pr,q)\}pr\{q\}$ to be valid for every $pr \in FPrg^{V,A}$ and every $q \in Pr^{V,A}$.

This binary composition

$$PC : FPrg^{V,A} \times Pr^{V,A} \xrightarrow{t} Pr^{V,A}$$

is simply referred to as preimage composition and is defined by the following formula ($q \in Pr^{V,A}$,$pr \in FPrg^{V,A}$ and $d \in {}^{V}A$):

$$PC(pr,q)(d) = \begin{cases} T, \text{ if } pr(d)\downarrow \text{ and } q(pr(d))\downarrow = T, \\ F, \text{ if } pr(d)\downarrow \text{ and } q(pr(d))\downarrow = F, \\ \text{undefined in other cases.} \end{cases} \quad (15)$$

It is easy to see that that this composition is *Glushkov prediction operation* (sequential execution of a function and a predicate) [8] and that it is also related (in the deterministic case) to the possibility/necessity operations of dynamic logic [9]. We called this composition (defined for partial predicates) as preimage predicate transformer composition in order to relate it to the weakest precondition predicate transformer.

The following equality can be easily proven using definitions of the preimage composition and Floyd-Hoare composition: $FH(p,pr,q) = p \to PC(pr,q)$.

Next, we define a unary composition TR of *predicate restriction on its truth domain* (*composition of deleting the falsity domain*), defined by the formula

$$TR(q)(d) = \begin{cases} T, \text{ if } q(d)\downarrow = T, \\ \text{undefined in other cases.} \end{cases} \quad (16)$$

At last, we specify a nowhere defined predicate $\bot_P$ such that $\bot_P(d)$ is undefined for any $d \in {}^{V}A$.

Thus, we obtain the following extended algebras (for different A) used to present more expressive assertions:

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

Logics constructed over such algebras are called *extended logics*.

Now we define a syntactical component of the constructed logics: their languages and inference systems.

3. INFERENCE SYSTEMS FOR FLOYD-HOARE LOGIC WITH PARTIAL PREDICATES

The language of logic as well as interpretation mappings for formulas emerge naturally from presented algebras (a semantic component of logics).

For a given sets of function symbols Fs, predicate symbols Ps, and variables V the sets of formulas $Fr(Ps,Fs,V)$ and terms $Tr(Ps,Fs,V)$ are inductively defined as follows:

- if $P \in Ps$ then $P \in Fr(Ps,Fs,V)$;
- if $\Phi,\Psi \in Fr(Ps,Fs,V)$ then $\Phi \vee \Psi, \Phi \wedge \Psi, \Phi \to \Psi$, $\neg\Phi \in Fr(Ps,Fs,V)$;
- if $\Phi \in Fr(Ps,Fs,V)$ and $v \in V$ then $\exists v\Phi \in Fr(Ps,Fs,V)$;
- if $P \in Ps$, $t_1,\ldots,t_n \in Tr(Ps,Fs,V)$ and $v_1,\ldots,v_n \in V$ are distinct variables then $S_P^{v_1,\ldots,v_n}(P,t_1,\ldots,t_n) \in Fr(Ps,Fs,V)$;
- if $F \in Fs$ then $F \in Tr(Ps,Fs,V)$;
- if $v \in V$ then ${}'v \in Tr(Ps,Fs,V)$;
- if $F \in Fs$, $t_1,\ldots,t_n \in Tr(Ps,Fs,V)$ and $v_1,\ldots,v_n \in V$ are distinct variables then $S_F^{v_1,\ldots,v_n}(F,t_1,\ldots,t_n) \in Tr(Ps,Fs,V)$.

Formulas and terms specify quasiary predicates and ordinary functions respectively.

Assertions are expressions of the form $\{p\}pr\{q\}$ (or $FH(p,pr,q)$) where $p,q \in Fr(Ps,Fs,V)$, and pr is a program.

Let us recall that in this paper we use the same notation for predicates and formulas. This is explained by the fact that for extensional completeness any predicate can be considered as pre- or postcondition.

A formula interpretation mapping is defined by an algebra $EA(V,A)$ and interpretations of function and predicate symbols in $EA(V,A)$. We denote an arbitrary interpretation as J [3].

We will use the following notations to give the definition of validity of the formula and of logical consequence:

- $p^T = \{d \mid p(d)\downarrow = T\}$ to denote the truth domain of predicate p;
- $p^F = \{d \mid p(d)\downarrow = F\}$ to denote the falsity domain of predicate p;
- p_J to denote the predicate that corresponds to the formula p under the interpretation J. For simplicity's sake we often omit J when it is clear from the context.

Validity of the formulas is considered as irrefutability. In other words, for every interpretation J the falsity domain of the respective predicate is empty:

$$\models p \Leftrightarrow p_J^F = \emptyset \text{ for every } J \quad (17)$$

Having definition of validity we will define $p \models q$ as $\models p \to q$. Also we need two special logical consequence relations. They will be used in specifying constraints of the inference systems:

- $p \models_T q \Leftrightarrow p_J^T \subseteq q_J^T$ for every interpretation J;
- $p \models_F q \Leftrightarrow q_J^F \subseteq p_J^F$ for every interpretation J.

These definitions also give examples that there can be more than one adequate logical consequence relation.

More detailed explanations are given in [3].

In order to apply a logic for verification, one should define an *inference system*.

The inference system for the language WHILE that was presented for classical Floyd-Hoare logic with total predicates [4] is not sound for the case of partial predicates. Therefore additional constraints have to be added to achieve a sound inference system. It could be done in several ways thus yielding different inference systems. Validity and completeness of such inference systems should be studied.

To denote that a formula p is derived in an inference system *IS* the notation $\vdash_{IS} p$ is used. The subscript is omitted when it is clear from the context.

Inference system is said to be *sound* iff every derived formula is valid, formally $\vdash p \Rightarrow \models p$ for every formula p.

Inference system is said to be *complete* iff every valid formula can be derived, formally $\models p \Rightarrow \vdash p$ for every formula p.

Completeness can be treated in extensional or intentional approaches. In the extensional approach pre- and postconditions can be arbitrary predicates. In the intentional approach pre- and postconditions should be presented by formulas of a given language.

Classical inference system *CI* for the language WHILE [4] (presented here in semantic form) is the following:

R_AS $\{S^x_P(p,h)\}AS^x(h)\{p\}$

R_SKIP $\{p\}\ id\ \{p\}$

R_SEQ $\frac{\{p\}f\ \{q\},\{q\}\ g\ \{r\}}{\{p\}\ f \bullet g\ \{r\}}$

R_IF $\frac{\{r\wedge p\}\ f\ \{q\},\{\neg r\wedge p\}g\ \{q\}}{\{p\}\ IF(r,f,g)\ \{q\}}$

R_WH $\frac{\{r\wedge p\}\ f\ \{p\}}{\{p\}\ WH(r,f)\ \{\neg r\wedge p\}}$

R_CONS $\frac{\{p'\}\ f\ \{q'\}}{\{p\}\ f\ \{q\}}, p \to p', q' \to q$

Such inference system is sound and extensionally complete for total predicates, but for partial predicates it is not sound. Rules *R_SEQ*, *R_WH*, and *R_CONS* do not guarantee a valid derivation from valid premises. One of the solutions of the problem is introduction of additional constraints for the mentioned rules [3].

Inference system *AC* with added constraints is presented below:

R_AS' $\{S^x_P(p,h)\}\ AS^x(h)\ \{p\}$

R_SKIP' $\{p\}\ id\ \{p\}$

R_SEQ' $\frac{\{p\}\ f\ \{q\},\{q\}\ g\ \{r\}}{\{p\}\ f \bullet g\ \{r\}}, p \models PC(f \bullet g, r)$

R_IF' $\frac{\{r\wedge p\}\ f\ \{q\},\{\neg r\wedge p\}g\ \{q\}}{\{p\}\ IF(r,f,g)\ \{q\}}$

R_WH' $\frac{\{r\wedge p\}\ f\ \{p\}}{\{p\}\ WH(r,f)\ \{\neg r\wedge p\}}, p \models PC(WH(r,f), \neg r \wedge p)$

R_CONS' $\frac{\{p'\}\ f\ \{q'\}}{\{p\}\ f\ \{q\}}, p \models_T p', q' \models_F q$

Inference system *AC* with added constraints is sound but constraints are rather complicated. Therefore simplifications of constraints are required. One of such simplifications stems from the following observation for properties of assertion validity for total predicates. In this case $\models \{p\}f\{q\}$ implies $p \models_T PC(f,q)$ and $p \models_F PC(f,q)$ [3]. This observation permits to consider a special class of assertions satisfying only the property $p \models_T PC(f,q)$. Such assertions are called *T-increasing assertions* because the truth domain of p is included in the preimage of the truth domain of q under f. Note that for partial predicates $p \models_T PC(f,q)$ implies $\models \{p\}f\{q\}$.

The dual class of *F-decreasing assertions* consists of assertions having the property $p \models_F PC(f,q)$.

It is important to admit that all rules of the classical inference system *CI* except the rule *R_CONS* preserve the class of T-increasing assertions. Moreover, this property grants satisfaction of the constraints for rules *R_SEQ'* and *R_WH'* of the inference system *AC*. Thus, constraints can be omitted if we modify the rule *R_CONS*. One of the inference systems that can be obtained in such manner is the following [3]:

R_AS $\{S^x_P(p,h)\}AS^x(h)\{p\}$

R_SKIP $\{p\}\ id\ \{p\}$

R_SEQ $\frac{\{p\}f\ \{q\},\{q\}\ g\ \{r\}}{\{p\}\ f \bullet g\ \{r\}}$

R_IF $\frac{\{r\wedge p\}\ f\ \{q\},\{\neg r\wedge p\}g\ \{q\}}{\{p\}\ IF(r,f,g)\ \{q\}}$

R_WH $\frac{\{r\wedge p\}\ f\ \{p\}}{\{p\}\ WH(r,f)\ \{\neg r\wedge p\}}$

R_CONS" $\frac{\{p'\}\ f\ \{q'\}}{\{p\}\ f\ \{q\}}, p \models_T p', q' \models_T q$

This inference system is called a *T-increasing system* and is denoted by *TI*. There is another sound inference system based on the property $p \models_F PC(f,q)$ that is dual to the above mentioned system *TI*.

Now we consider properties of the constructed inference systems concentrating on their completeness.

4. EXTENSIONAL AND INTENSIONAL COMPLETENESS OF INFERENCE SYSTEMS

Let us start with the inference system *TI* which has constraints only for rule *R_CONS"*.

First, we should admit that there are Floyd-Hoare assertions that are valid but do not belong to the class of T-increasing assertions. For example, consider a predicate p that is true on every data, and a predicate q that is undefined on every data. The triple $\{p\}\ id\ \{q\}$ is valid, but $p \not\models_T PC(id,q)$. This example demonstrates that the inference system *TI* is incomplete for the general class of valid assertions. Still, for T-increasing assertions the following result will be proved.

Theorem 4.1. *The inference system TI is extensionally complete for the class of T-increasing assertions.*

To prove this theorem we will first prove $\vdash_{TI} \{PC(pr,q)\}\ pr\ \{q\}$ for every program pr and every predicate q by induction over the program structure.

Base of the induction.

For $\vdash_{TI} \{PC(AS^x(h),q)\}\ AS^x(h)\ \{q\}$ we have to show that $PC(AS^x(h),q) = S^x_P(q,h)$. Thus the rule *R_AS* will give the required result.

To prove this equality we use the definition of preimage composition

$$PC(AS^x(h),q)(d) = \begin{cases} T, \text{ if } q(AS^x(h)(d)) \downarrow = T, \\ F, \text{ if } q(AS^x(h)(d)) \downarrow = F, \\ \text{undefined in other cases.} \end{cases}$$

and by the definition of assignment composition we obtain

$$PC(AS^x(h),q)(d) = \begin{cases} T, \text{ if } q(d\nabla[x \mapsto h(d)]) \downarrow = T, \\ F, \text{ if } q(d\nabla[x \mapsto h(d)]) \downarrow = F, \\ \text{undefined in other cases.} \end{cases}$$

This gives $PC(AS^x(h),q)(d) = q(d\nabla[x \mapsto h(d)])$. But also by the definition of superposition composition $S^x_P(q,h) = q(d\nabla[x \mapsto h(d)])$. Hence, we obtain

$$PC(AS^x(h),q) = S^x_P(q,h).$$

Therefore $\vdash_{TI} \{PC(AS^x(h),q)\}\ AS^x(h)\ \{q\}$.

For $\vdash_{TI} \{PC(id,q)\}\ id\ \{q\}$ the proof is obvious. If we look at the definition of preimage composition, it is not hard to check that $PC(id,q) = q$ for an arbitrary predicate q. Thus, $\vdash_{TI} \{q\}\ id\ \{q\}$ follows by rule *R_SKIP*.

Inductive step.

Consider the case of sequential execution. We need to prove that $\vdash_{TI} \{PC(pr_1 \bullet pr_2,q)\}\ pr_1 \bullet pr_2\ \{q\}$ given

$$\vdash_{TI} \{PC(pr_2,q)\}\ pr_2\ \{q\}$$

and

$$\vdash_{TI} \{PC(pr_1,PC(pr_2,q))\}\ pr_1\ \{PC(pr_2,q)\}.$$

Applying the rule *R_SEQ* to both premises we will obtain $\vdash_{TI} \{PC(pr_1,PC(pr_2,q))\}\ pr_1 \bullet pr_2\ \{q\}$.

If we show that

$$PC(pr_1,PC(pr_2,q))^T = PC(pr_1 \bullet pr_2,q)^T$$

then

$$PC(pr_1 \bullet pr_2,q) \models_T PC(pr_1,PC(pr_2,q)).$$

After that we can apply the rule *R_CONS"* to $\vdash_{TI} \{PC(pr_1,PC(pr_2,q))\}\ pr_1 \bullet pr_2\ \{q\}$ and get the following assertion proved:

$$\vdash_{TI} \{PC(pr_1 \bullet pr_2,q)\}\ pr_1 \bullet pr_2\ \{q\}.$$

If $d \in PC(pr_1,PC(pr_2,q))^T$ then $pr_1(d) \downarrow$ and $PC(pr_2,q)(pr_1(d)) \downarrow = T$.

Thus, $pr_2(pr_1(d)) \downarrow$ and $q(pr_2(pr_1(d))) \downarrow = T$. Using the definition of sequential execution composition this can be rewritten as $pr_1 \bullet pr_2(d) \downarrow$ and $q(pr_1 \bullet pr_2(d)) \downarrow = T$. Hence $d \in PC(pr_1 \bullet pr_2,q)^T$.

If $d \in PC(pr_1 \bullet pr_2,q)^T$ then $pr_1 \bullet pr_2(d) \downarrow$ and $q(pr_1 \bullet pr_2(d)) \downarrow = T$, or $pr_2(pr_1(d)) \downarrow$ and $q(pr_2(pr_1(d))) \downarrow = T$.

Hence, $PC(pr_2,q)(pr_1(d)) \downarrow = T$ and $pr_1(d) \downarrow$. Thus, $d \in PC(pr_1,PC(pr_2,q))^T$.

We have shown that

$$PC(pr_1,PC(pr_2,q))^T = PC(pr_1 \bullet pr_2,q)^T,$$

hence,

$$\vdash_{TI} \{PC(pr_1 \bullet pr_2,q)\}\ pr_1 \bullet pr_2\ \{q\}.$$

In the case of the conditional composition, $\vdash_{TI} \{PC(IF(r,pr_1,pr_2),q)\}\ IF(r,pr_1,pr_2)\ \{q\}$ have to be proved, given $\vdash_{TI} \{PC(pr_1,q)\}\ pr_1\ \{q\}$ and $\vdash_{TI} \{PC(pr_2,q)\}\ pr_2\ \{q\}$. If we show that $r \wedge PC(IF(r,pr_1,pr_2),q) \models_T PC(pr_1,q)$ together with $\neg r \wedge PC(IF(r,pr_1,pr_2),q) \models_T PC(pr_2,q)$, then using *R_CONS"* rule we obtain the following $\vdash_{TI} \{r \wedge PC(IF(r,pr_1,pr_2),q)\}\ pr_1\ \{q\}$ and also $\vdash_{TI} \{\neg r \wedge PC(IF(r,pr_1,pr_2),q)\}\ pr_2\ \{q\}$. Using *R_IF* with these derived assertions as premises we will get the needed result, which means

$$\vdash_{TI} \{PC(IF(r,pr_1,pr_2),q)\}\ IF(r,pr_1,pr_2)\ \{q\}.$$

Let us prove

$$\neg r \wedge PC(IF(r,pr_1,pr_2),q) \models_T PC(pr_2,q).$$

If $d \in (\neg r \wedge PC(IF(r,pr_1,pr_2),q))^T$ then $r(d) \downarrow = F$, $IF(r,pr_1,pr_2)(d) \downarrow$, together with $q(IF(r,pr_1,pr_2)(d)) \downarrow = T$.

Since $r(d) \downarrow = F$ and $IF(r,pr_1,pr_2)(d) \downarrow$ it follows that $IF(r,pr_1,pr_2)(d) \downarrow = pr_2(d)$. Thus $pr_2(d) \downarrow$ and $q(pr_2(d)) \downarrow = T$. In other words, $PC(pr_2,q)(d) \downarrow = T$, which means $d \in PC(pr_2,q)^T$. Thus,

$$\neg r \wedge PC(IF(r,pr_1,pr_2),q) \models_T PC(pr_2,q).$$

The property $r \wedge PC(IF(r,pr_1,pr_2),q) \models_T PC(pr_1,q)$ can be proved in the same way. Having both premises proved we obtain the needed result

$$\vdash_{TI} \{PC(IF(r,pr_1,pr_2),q)\}\ IF(r,pr_1,pr_2)\ \{q\}.$$

Let us show that $\vdash_{TI} \{PC(WH(r,pr),q)\}WH(r,pr)\{q\}$ for every pr, q, r. Let $p = PC(WH(r,pr),q)$. By induction we have that $\vdash_{TI} \{PC(pr,p)\}\ pr\ \{p\}$. If we show that $r \wedge p \models_T PC(pr,p)$ then by *R_CONS"* we will have $\vdash_{TI} \{r \wedge p\}\ pr\ \{p\}$. After that by *R_WH*, we get $\vdash_{TI} \{p\}\ WH(r,pr)\ \{\neg r \wedge p\}$. If $\neg r \wedge p \models_T q$ then $\vdash_{TI} \{p\}\ WH(r,pr)\ \{q\}$ can be obtained by the rule *R_CONS"*, what was required to be proved.

Let us first prove $\neg r \wedge p \models_T q$. For an arbitrary d if $(\neg r \wedge p)(d) \downarrow = T$ then $r(d) \downarrow = F$. Thus $WH(r,pr)\ (d) \downarrow = d$ and $q(d) \downarrow = q(WH(r,pr)\ (d))$. Also from $p(d) \downarrow = T$ and $p = PC(WH(r,pr),q)$ we have $q(WH(r,pr)\ (d)) \downarrow = T$. Hence $q(d) \downarrow = T$ and $\neg r \wedge p \models_T q$.

To prove $r \wedge p \models_T PC(pr,p)$ we need to recall that $p = PC(WH(r,pr),q)$. If $(r \wedge p)(d) \downarrow = T$ then $p(d) \downarrow = T$ which gives $q(WH(r,pr)(d)) \downarrow = T$. Together with $r(d) \downarrow = T$ the definition of cyclic composition gives that there exists such $d' = pr(d)$ that $WH(r,pr)(d') \downarrow = WH(r,pr)(d)$ and $q(WH(r,pr)(d')) \downarrow = T$. By the definition of preimage composition we have that $p(d') = PC(WH(r,pr),q)(d') \downarrow = T$. Since $p(d') \downarrow = T$ and

$d' = pr(d)$ we have $p(pr(d)) \downarrow = T$ and $PC(pr,p)(d) \downarrow = T$. Hence, $r \wedge p \models_T PC(pr,p)$.

We proved both needed logical consequences thus $\vdash_{TI} \{PC(WH(r,pr),q)\}\ WH(r,pr)\ \{q\}$.

We have considered all compositions, so, $\vdash_{TI} \{PC(pr,q)\}\ pr\ \{q\}$ for every pr and q.

Based on this property we will prove $\vdash_{TI} \{q\}\ pr\ \{q\}$ for any T-increasing assertion $\{p\}\ pr\ \{q\}$. To do this we consider predicate $q'' = TR(q)$. It can be shown that $q'' \models_T q$ since $q''^T = q^T$. Also $p \models_T PC(pr,q'')$, because $p \models_T PC(pr,q)$ and $q''^T = q^T$. Previously it was proven that $\vdash_{TI} \{PC(pr,q'')\}\ pr\ \{q''\}$. By *R_CONS"* we obtain $\vdash_{TI} \{p\}\ pr\ \{q\}$. Thus, the inference system *TI* is extensionally complete for the class of T-increasing assertions.

Now let us consider completeness of the inference system *AC* with added constraints.

Theorem 4.2. *The inference system AC is extensionally complete for the general class of assertions.*

To prove the theorem we will first prove by induction over the program structure the following property:

$\vdash_{AC} \{PC(pr,q)\}\ pr\ \{q\}$ for every pr and q.

Base of the induction.

Rules *R_AS'* and *R_AS* are identical as well as rules *R_SKIP'* and *R_SKIP*. Thus, the proof is similar to the proof for the previous inference system *TI* and is omitted here. So, $\vdash_{AC} \{PC(AS^x(h),q)\}\ AS^x(h)\ \{q\}$ and $\vdash_{AC} \{PC(id,q)\}\ id\ \{q\}$.

Inductive step.

Rules *R_IF* and *R_IF'* are identical. If $q = q'$, rules *R_CONS'* and *R_CONS"* are also identical. Thus the proof for the rule *R_IF'* is similar to the proof for the rule *R_IF* for the previous inference system and is also omitted. This gives us

$$\vdash_{AC} \{PC(IF(r,pr_1,pr_2),q)\}\ IF(r,pr_1,pr_2)\ \{q\}$$

Rules *R_SEQ* and *R_SEQ'* differ only in the added constraint, but with $PC(pr_1,PC(pr_2,q))^T = PC(pr_1 \bullet pr_2,q)^T$ that was proven earlier, we have needed property $PC(pr_1,PC(pr_2,q)) \models PC(pr_1 \bullet pr_2,q)$ and the constrain is satisfied given $\vdash_{AC} \{PC(pr_2,q)\}\ pr_2\ \{q\}$ and $\vdash_{AC} \{PC(pr_1,PC(pr_2,q))\}\ pr_1\ \{PC(pr_2,q)\}$ as premises. Rest of the proof for this rule is similar to the proof for rule *R_SEQ* and is omitted. Hence, $\vdash_{AC} \{PC(pr_1 \bullet pr_2,q)\}\ pr_1 \bullet pr_2\ \{q\}$.

Let us show that $\vdash_{AC} \{PC(WH(r,pr),q)\}\ WH(r,pr)\ \{q\}$ for every pr, q, r. Let $p = PC(WH(r,pr),q)$. Consider the following predicate:

$$p'(d) = \begin{cases} T, \text{ if } p(d) \downarrow = T, \\ F, \text{ otherwise.} \end{cases}$$

We have that $\vdash_{AC} \{PC(pr,p')\}\ pr\ \{p'\}$. If we show that $r \wedge p' \models_T PC(pr,p')$ then by *R_CONS'* we will have $\vdash_{AC} \{r \wedge p'\}\ pr\ \{p'\}$. After that by *R_WH'*, if $p' \models PC(WH(r,pr),\neg r \wedge p')$, we get

$$\vdash_{AC} \{p'\}\ WH(r,pr)\ \{\neg r \wedge p'\}.$$

If $\neg r \wedge p' \models_F q$ and $p \models_T p'$ then by the rule *R_CONS'* $\vdash_{AC} \{p\}\ WH(r,pr)\ \{q\}$ can be obtained, what was needed to be proved.

Proof of the $p \models_T p'$ is obvious by the definition of p'.

To prove $r \wedge p' \models_T PC(pr,p')$ we need to recall that $p = PC(WH(r,pr),q)$ and $p^T = p'^T$. If $(r \wedge p')(d) \downarrow = T$ then $p'(d) \downarrow = p(d) = T$ that gives $q(WH(r,pr)(d)) \downarrow = T$. Together with $r(d) \downarrow = T$ the definition of cyclic composition gives that there exists such a state $d' = pr(d)$ that $WH(r,pr)(d') \downarrow = WH(r,pr)(d)$ and $q(WH(r,pr)(d')) \downarrow = T$. By the definition of preimage condition composition we have that $p'(d') \downarrow = p(d') = PC(WH(r,pr),q)(d') = T$. By $p'(d') \downarrow = T$ and $d' = pr(d)$ we have $PC(pr,p')(d) \downarrow = T$. Hence $r \wedge p' \models_T PC(pr,p')$.

Let us prove $\neg r \wedge p' \models_F q$. If $q(d) \downarrow = F$ then there can be three cases.

If $r(d) \downarrow = T$ then $(\neg r \wedge p')(d) \downarrow = F$.

If $r(d) \downarrow = F$ then $p(d) = PC(WH(r,pr),q)(d) \downarrow = F$. Thus $p'(d) = p(d) \downarrow = F$ and $(\neg r \wedge p')(d) \downarrow = F$.

If $r(d) \uparrow$ then $p(d) = PC(WH(r,pr),q)(d) \uparrow$. Thus $p'(d) \downarrow = F$ and $(\neg r \wedge p')(d) \downarrow = F$.

In all cases $(\neg r \wedge p')(d) \downarrow = F$. Hence, $\neg r \wedge p' \models_F q$.

Property $p' \models PC(WH(r,pr),\neg r \wedge p')$ is obvious. If $p'(d) \downarrow = T$ then $p(d) \downarrow = T$ by the definition of p'. Thus there exists such d' that $WH(r,pr)(d) \downarrow = d'$ and $q(d') = T$, because $p = PC(WH(r,pr),q)$. The definition of the cyclic composition gives us that $r(d') \downarrow = F$ and $WH(r,pr)(d') \downarrow = d'$. Since $WH(r,pr)(d') \downarrow = d'$ and $q(d') = T$ we have $p(d') \downarrow = T = PC(WH(r,pr),q)(d')$. Hence $p'(d') \downarrow = T$ and $(\neg r \wedge p')(d') \downarrow = T$. And with $WH(r,pr)(d) \downarrow = d'$ we have $PC(WH(r,pr),\neg r \wedge p')(d) \downarrow = T$. This proves $p'^T \subseteq PC(WH(r,pr),\neg r \wedge p')^T$ that grants the needed logical consequence.

We proved all needed logical consequences, thus $\vdash_{AC} \{PC(WH(r,pr),q)\}\ WH(r,pr)\ \{q\}$.

We have considered all compositions, therefore $\vdash_{AC} \{PC(pr,q)\}\ pr\ \{q\}$ for every program pr and predicate q.

If we have some valid assertion $\models \{p\}\ pr\ \{q\}$, we can find such q' that $p \models_T PC(pr,q')$ and $q' \models_F q$. Then $\vdash_{AC} \{PC(pr,q')\}\ pr\ \{q'\}$, and $\vdash_{AC} \{p\}\ pr\ \{q\}$ by *R_CONS'*. Consider q' such that:

$$q'(d) = \begin{cases} T, \text{ if } p(pr^{-1}(d)) \downarrow = T, \\ F, \text{ if } q(d) \downarrow = F, \\ \text{undefined in other cases.} \end{cases}$$

It is not hard to show that $q^F \subseteq q'^F$ and $p^T \subseteq PC(pr,q')^T$. Property $\models \{p\}\ pr\ \{q\}$ implies that such a predicate exists because the validity of assertion implies that there is no such d that $FH(p,pr,q)(d) \downarrow = F$. By the definition of Floyd-Hoare composition, $FH(p,pr,q)(d) \downarrow = F$ gives $p(d) \downarrow = T$ and $q(pr(d)) \downarrow = F$. This implies that there is no such $d' = pr(d)$ that $q'(d') \downarrow = T$ and $q'(d') \downarrow = F$. Thus, the extensional completeness of the inference system *AC* is proved.

Now let us consider intensional completeness of inference systems *TI* and *AC*. Here intensional completeness is understood in the following way: 1) predicates in assertions, inference rules, and constraints are represented as

formulas of the class $Fr(Ps,Fs,V)$; 2) constraints are represented as formulas derived in the inference system under consideration. For example, the constraints for the rule *R_CONS"* should be presented in the form $p \vdash_{TI} p'$ and $q' \vdash_{TI} q$.

For the inference system *TI* there are two main difficulties in proving its intensional completeness: 1) formulas of the form $PC(pr,p)$ and $TR(p)$ should be reduced to formulas of the class $Fr(Ps,Fs,V)$; 2) we should have intensionally complete inference system to prove formulas of the form $p \vdash_{TI} p'$.

As to the first difficulty we should admit that the required reduction of $PC(pr,p)$ is possible for acyclic programs, but the reduction for programs with cycles requires more expressive languages with, say, the least fixed point operators. Reduction of $TR(p)$ is trivial because $TR(p) = p \vee \bot_P$. As to the second difficulty, the intensional completeness of the inference system in hand was proved in [10, 11].

Therefore we consider here only intensional completeness of *TI* for acyclic programs.

We have the following reductions:

$$PC(id,q) = q \tag{18}$$

$$PC(AS^x(h),q) = S^x(q,h) \tag{19}$$

$$PC(pr_1 \bullet pr_2,q) = PC(pr_1,PC(pr_2,q)) \tag{20}$$

$$\begin{aligned} PC(IF(r,pr_1,pr_2),q) = (r \to PC(pr_1,q)) \wedge \\ \wedge(\neg r \to PC(pr_2,q)) \wedge (r \to r) \end{aligned} \tag{21}$$

Correctness of the first three reductions was proved earlier. For the fourth formula we will prove that for every data d

$$\begin{aligned} PC(IF(r,pr_1,pr_2),q)(d) = ((r \to PC(pr_1,q)) \wedge \\ \wedge(\neg r \to PC(pr_2,q)) \wedge (r \to r))(d) \end{aligned}$$

Let us consider three cases, depending on the value of $r(d)$.

If $r(d) \downarrow = T$ then $(r \to r)(d) \downarrow = T$, with $(\neg r \to PC(pr_2,q))(d) \downarrow = T$ and the following $(r \to PC(pr_1,q))(d) = PC(pr_1,q)(d)$. Thus the right side is equal to $PC(pr_1,q)(d)PC(pr_1,q)(d)$. Also

$PC(IF(r,pr_1,pr_2),q)(d) = PC(pr_1,q)(d)$, since $IF(r,pr_1,pr_2)(d) = pr_1(d)$.

Hence both sides of the equality are equal to $PC(pr_1,q)(d)$.

In the case when $r(d) \downarrow = F$ the proof is similar.

If $r(d) \uparrow$ then $PC(IF(r,pr_1,pr_2),q)(d) \uparrow$ and $(r \to r)(d) \uparrow$. Expressions $(\neg r \to PC(pr_2,q))(d)$ and $(r \to PC(pr_1,q))(d)$ can be either true or undefined, depending on the values of $PC(pr_2,q)(d)$ and $PC(pr_1,q)(d)$ respectively. In either case by the right side of the equality will be undefined by the definition of the conjunction composition. But without the conjunct $(r \to r)$ in the case when $r(d) \uparrow$ the right side of the equality could be true, and not undefined as required.

Let us note that all proofs are correct for any interpretation J considered here implicitly. This permits us to formulate the following result.

Theorem 4.3. *The inference system TI is intensionally complete for the class of T-increasing assertions with acyclic programs.*

Intensional completeness of the inference system *AC* requires additional investigations.

5. RELATED WORK

The first attempts to use logical approach to define specification of the programs and reason about their properties were made by Glushkov [8], Floyd [1], and Hoare [2]. Viktor Glushkov developed algorithmic algebras based on three-valued logics while Robert Floyd and Tony Hoare were oriented to axiomatic systems with total predicates. Simple idea to represent program specifications as triples that consist of precondition, program, and postcondition turned out to be very powerful and found many implementations.

With time it appeared that in more and more cases data or predicates can be undefined. It became evident that partiality of predicates and functions have to be taken into consideration and proper formalisms should be developed. This give rise to the special three-valued logics, where the fact that predicate is undefined is represented by special third value. Numerous variants of such logics were presented and studied in the works of Lukasiewicz, Kleene, Bochvar and others.

At the same time evolution of the programming languages, emergence of the new features as pointers, dynamic typing caused the need of the tools to define more complex specifications. This led to the development of the numerous extensions of Floyd-Hoare logic that have proved to be effective basis for program verification. The most known and studied extensions of Floyd-Hoare logic are Dynamic logic and Separation logic.

Dynamic logic [9] is an attempt to redefine Floyd-Hoare logic using modalities that permit usage of program texts and specifications alongside. Floyd-Hoare assertion $\{p\}pr\{q\}$ can be substituted with formula $p \to [pr]q$ where $[pr]q$ means that program pr if it terminates has to satisfy formula q. It is not hard to check that if pr is single-valued (deterministic) this formula is equal to the preimage composition presented in this paper and it is also possible to use specifications and program texts together.

Separation logic [12] was introduced as a response to broad usage of heap and pointers in programming. New special implications provide terms to specify different properties of the heap. But only heap function that mapped memory addresses to values assumed to be partial. In all other aspects only total predicates are considered. The main goal was to introduce means to deal with pointers but general case of partial data is omitted.

Logics describing various properties of programs form a basis for formal software system development [13].

6. CONCLUSIONS

In this paper extensions of Floyd-Hoare logic for partial predicates have been presented. Different inference systems have been defined. Problems of extensional and intentional completeness have been studied for the introduced systems. Special cases of program logics have been considered for which completeness problems can be reduced to the similar problem in first-order predicate logics.

Further directions of research include extensions of constructed logics for more powerful classes of programs with structured data and compositions of parallel execution.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Ievgen Ivanov and anonymous reviewers for reading early versions of this paper and providing helpful feedback.

REFERENCES

[1] FLOYD, R.W.: Assigning meanings to programs, *Proceedings of the American Mathematical Society Symposia on Applied Mathematics*, vol. 19, pp. 19-31, 1967.

[2] HOARE, C.A.R.: An axiomatic basis for computer programming, *Communications of ACM*, issue 12, pp. 578-580,583; 1969.

[3] KRYVOLAP, A. –NIKITCHENKO, M. –SCHREINER, W.: Extending Floyd-Hoare logic for partial pre- and postconditions, In: V. Ermolayev et al. (eds.): ICTERI 2013, CCIS, vol. 412, pp. 355-378, Springer, Heidelberg, 2013

[4] NIELSON, H.R. –NIELSON, F.: *Semantics with applications: a formal introduction*, John Wiley & Sons Inc. 240 p., 1992.

[5] NIKITCHENKO, M.S. –SHKILNIAK, S.S.: *Mathematical logic and theory of algorithms*, Publishing house of Taras Shevchenko National University of Kyiv, Kyiv, 528 p., 2008, in Ukrainian.

[6] NIKITCHENKO, M.S. –TYMOFIEIEV, V.G.: Satisfiability in composition-nominative logics,*Central European Journal of Computer Science*, vol. 2, issue 3, pp. 194-213 (2012).

[7] DIJKSTRA, E.W.: *A discipline of programming*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[8] GLUSHKOV, V.M.: Automata theory and formal transformations of microprograms, *Cybernetics*, No. 5, pp. 3-10 (1965). In Russian.

[9] HAREL, D. –KOZEN, D. –TIURYN, J.: Dynamic logic, *Handbook of Philosophical Logic*, pp. 497-604, 1984.

[10] NIKITCHENKO, M.S. –SHKILNIAK, S.S.: Composition-nominative logics of quasiary predicates: semantic aspects. *Visnyk, Ser. Fiz.-Mat. Nauky, Kyiv Univ. im. Tarasa Shevchenka*, issue 4, pp. 165-172, 2012, in Ukrainian.

[11] SHKILNIAK, S.S.: The spectrum of sequent calculi of first-order composition-nominative logics, *Problems of Programming*, issue 3, pp. 22-37, 2013, in Ukrainian.

[12] REYNOLDS, J.C.: Separation logic: A logic for shared mutable data structures, *LICS'02*, pp.55-74, 2002.

[13] SANNELLA, D. –Tarlecki A.: *Foundations of Algebraic Specification and Formal Software Development*, Springer, 2012.

Received June ..., 2014, accepted September ..., 2014

BIOGRAPHY

Mykola (Nikolaj) Nikitchenko, Prof., Dr. is a professor at the Faculty of Cybernetics of Taras Shevchenko National University of Kyiv, Ukraine and the chairman of the Department of Theory and Technology of Programming of this university. His scientific research is focused on methodology of programming, formal methods, databases, mathematical logic and computability theory.

Andrii Kryvolap graduated (MSc) with distinction from the Faculty of Cybernetics of Taras Shevchenko National University of Kyiv, Ukraine in 2011. Now he is a PhD student at Taras Shevchenko National University of Kyiv. His scientific research is focused on formal methods, program logics and inference systems.