ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2009
Informatyka – Zeszyt 4

Agnieszka Makarec[1]

PROBABILISTIC AND NON-DETERMINISTIC SEMANTICS FOR ITERATIVE PROGRAMS

Abstract: In this paper probabilistic and non-deterministic programs are considered on the ground of logic of programs. We are interested in dependencies between nondeterministic and probabilistic interpretation of a program. The formal definitions of probabilistic and non-deterministic semantics are the starting point for our considerations. The emphasis is on differences in expressibility the halting property in probabilistic and non-deterministic logic of programs.

Keywords: non-deterministic program, probabilistic program, non-deterministic computation, probabilistic computation

1. Introduction

There are many reasons that make the problem of non-deterministic and probabilistic constructions an important field of study: average case analysis, randomized algorithms, distributed systems, concurrency, reliability, security. There were proposed several approaches to model of non-deterministic and random choice of the control in programming languages (cf. [2,3,4,6,7]). Non-determinism can be used to abstract over probabilities. It is a valuable tool that helps us understand the expressiveness of programming language constructs. However, the input to a program and the output may be described in terms of a probability distribution, in which case the average case behavior of the program can be formally analyzed.

2. Non-deterministic and probabilistic interpretations of random assignments

The considerations of the presented paper will be formulated in terms of logics of programs. We shall compare properties of iterative programs expressible in non-deterministic and probabilistic versions of algorithmic logic (cf. [1,8]). To compare these two approaches we assume that computations of probabilistic and non-deterministic programs are realized in a fixed algebraic structure for language L_{alg},

[1] Faculty of Computer Science, Bialystok Technical University, Białystok

containing assignments of the form $x :=?$ and the construction *either ...or ...* . Probabilistic and non-deterministic interpretation of assignments $x :=?$ should be *consistent* in a natural sense: an element is probabilistically generated with a positive probability if and only if non-deterministic generation of this element is possible.

The concepts of them seem to be essentially different but some dependencies can be formulated and proved. Namely, if in probabilistic interpretation, program K halts with positive probability, then in the non-deterministic interpretation the formula expressing the possibility of halting is valid. In the present paper we restrict ourselves to finite interpretations because of the efficiency of verification of programs properties.

The algorithmic language L_{alg}, being an extension of first-order language L, is an abstract programming language of iterative programs.

There are three kinds of well-formed expressions in L_{alg}: terms, formulas and programs. The alphabet of the language L_{alg} contains enumerable sets of predicates Ψ, functors Φ and a countable set V_1 of individual variables. The set T of terms and the set F of first-order formulas are defined as in the classical logic. The set Π of iterative programs is the least set of expressions containing assignments $x :=?, x := \tau$, where $x \in V_1, \tau \in V_1$, and closed under the program constructions: *begin* M; N *end* , *if* γ *then* M *else* N , *while* γ *do* M, *either* M *or* N.

3. Probabilistic Algorithmic Logic

In this subsection we describe the syntax and semantics of language L_{prob}, which is an extension of L_{alg} (such that $L \subset L_{alg} \subset L_{prob}$). The syntax and semantics of language L_{prob} derives from Probabilistic Algorithmic Logic PrAL [1].

3.1 Syntax of L_{prob}

The alphabet of the language L_{prob} contains:
Ψ – an enumerable set of predicates (the equality sign = is distinguished), which includes subset Ψ_R of arithmetical predicates and Ψ_1 of non arithmetical predicates,
Φ – an enumerable set of functors, which includes subset $\Phi_R = \{+, -, *, /\}$ of arithmetical functors and subset Φ_1 of non arithmetical functors,
V – an countable set of variables, which includes subset V_1 of individual variables and subset V_R of real variables,
C – a finite set of real constants.
L_{prob} is a two-sorted language with the sets of terms T and T_R , and the set of all formulas F_{prob}.
The set T_R of arithmetical terms is the least set of expressions such that:

i. if $\alpha \in F$ and $K \in \Pi$ then $P(\alpha), P(K\alpha) \in T_R$,
ii. $V_R \subset T_R$,
iii. if $\varphi \in \Phi_R$ is an j-argument functor and $\tau_1, \ldots, \tau_j \in T_R$ then $\varphi(\tau_1, \ldots, \tau_j) \in T_R$

The set F_{prob} is the least set of expressions such that:

i. if $\alpha \in F$ and $K \in \Pi$ then $K\alpha \in F_{prob}$,
ii. if $\alpha, \beta \in F_{prob}$ then $\alpha \vee \beta, \alpha \wedge \beta, \alpha \Rightarrow \beta, \neg\alpha \in F_{prob}$,
iii. if $\xi \in \Psi_R$ is an j-argument predicate and $\tau_1, \ldots, \tau_j \in T_R$ then $\xi(\tau_1, \ldots, \tau_j) \in F_{prob}$.

The set F_0 of *open formulas* is the least set of expressions such that:

i. if $\sigma \in \Psi_1$ is an j-argument predicate and $\tau_1, \ldots, \tau_j \in T$ then $\sigma(\tau_1, \ldots, \tau_j) \in F_0$,
ii. if $\alpha, \beta \in F_0$ then $\alpha \vee \beta, \alpha \wedge \beta, \alpha \Rightarrow \beta, \neg\alpha \in F_0$.

3.2 Semantics of L_{prob}

The interpretation of the language L_{prob} will be understood as the triple $\langle \Im, \rho, p \rangle$, where $\Im$ is a structure for L with the universe $U = \{u_1, \ldots, u_r\}$ and $\rho : U \to [0,1]$ is the fixed probabilistic distribution on the set U connected with a random assignment $x :=?$, such that

$$\rho(u_i) = p_i, \sum_{i=1,\ldots,r} p_i = 1. \tag{1}$$

The number p, satisfying $0 < p < 1$, corresponds to the probability of choosing the subprogram M_1 as the result of the realization of the construction *either* M_1 *or* M_2.

By a *valuation* of individual variables we mean a mapping $\omega : V_1 \to U$, where $V_1 = \{x_1, x_2, \ldots\}$. By a valuation of real variables we mean a mapping $\omega_R : V_R \to R$. Let us denote by $K(x_1, \ldots, x_n)$ the probabilistic program with all variables among $\{x_1, \ldots, x_n\}$. We will write it simply K when no confusion can arise. Each *state* of program K is described by the valuation $\omega = (\omega(x_1), \ldots, \omega(x_n))$. We will denote by $W = \{\omega_1, \ldots, \omega_l\}, l = r^n$ the set of all possible valuations of variables from $\{x_1, \ldots, x_n\}$. Fact that the valuation ω in the structure $\langle \Im, \rho, p \rangle$ satisfies the formula α will be denoted by $\langle \Im, \rho, p \rangle, \omega \models \alpha$ (or shortly $\omega \models \alpha$).

The interpretation of open formulas and terms in the language L_{prob} is classical and it was presented in [1,5].
The element $\tau_{\langle \Im, \rho, p \rangle}(\omega)$ of U is called *the value of the term* τ *in the structure* $\langle \Im, \rho, p \rangle$ *at the valuation* ω and it will be shortly denoted by $\tau(\omega)$ (when no confusion can arise).

3.3 Computational semantics of L_{prob}

By a *configuration* in the structure $\langle \Im, \rho, p \rangle$ we understand any ordered triple of the form $s = \langle \omega; q; \overrightarrow{K} \rangle$, where ω is a valuation, q is a probability of occurring that valuation, $\overrightarrow{K} = (\mathrm{K}_1, \ldots, \mathrm{K}_m)$ is a finite sequence of programs, which are executed in order as they are written (the sequence may be empty).

A *computation* of a probabilistic program K with an input valuation ω, which appears with probability q, we mean as the sequence of configurations satisfying the following conditions.

I. The first element of this sequence is in the form of $s_1 = \langle \omega; q; \mathrm{K} \rangle$.

II. If i-th configuration $s_i = \langle \omega'; q'; \mathrm{K}_1, \ldots, \mathrm{K}_m \rangle$ of the sequence is determined and q' denotes a probability of occurring valuation ω' then s_{i+1} is the next configuration and:

   i. If K_1 is in the form of $x := \tau$, then $s_{i+1} = \langle \omega''; q'; \mathrm{K}_2, \ldots, \mathrm{K}_m \rangle$, where

$$\omega''(y) = \begin{cases} \tau(\omega') \text{ iff } y = x, \\ \omega'(y) \text{ iff } y \neq x. \end{cases} \tag{2}$$

   ii. If K_1 is in the form of $x :=?$, then $s_{i+1} = \langle \omega''; q' \cdot \rho(u); \mathrm{K}_2, \ldots, \mathrm{K}_m \rangle$, where $\rho(u)$ denotes probability assignment of an element $u \in U$ to a variable x and

$$\omega''(y) = \begin{cases} u \quad \text{iff } y = x, \\ \omega'(y) \text{ iff } y \neq x. \end{cases} \tag{3}$$

   iii. If K_1 is in the form of *either* M *or* N, then

$$s_{i+1} = \langle \omega'; q' \cdot p; \mathrm{M}, \mathrm{K}_2, \ldots, \mathrm{K}_m \rangle \text{ or } s_{i+1} = \langle \omega'; q' \cdot (1-p); \mathrm{N}, \mathrm{K}_2, \ldots, \mathrm{K}_m \rangle \tag{4}$$

   and p denotes the probability of choosing the subprogram M.

   iv. If K_1 is in the form of *begin* M; N *end*, then

$$s_{i+1} = \langle \omega'; q'; \mathrm{M}, \mathrm{N}, \mathrm{K}_2, \ldots, \mathrm{K}_m \rangle. \tag{5}$$

   v. If K_1 is in the form of *if* γ *then* M *else* N, then

$$s_{i+1} = \begin{cases} \langle \omega'; q'; M, K_2, \ldots, K_m \rangle \text{ iff } \omega' \models \gamma, \\ \langle \omega'; q'; N, K_2, \ldots, K_m \rangle \text{ otherwise.} \end{cases} \tag{6}$$

   vi. If K_1 is in the form of *while* γ *do* M, then

$$s_{i+1} = \begin{cases} \langle \omega'; q'; \mathrm{M}, \textit{while } \gamma \textit{ do } \mathrm{M}, \mathrm{K}_2, \ldots, \mathrm{K}_m \rangle \text{ iff } \omega' \models \gamma, \\ \langle \omega'; q'; \mathrm{K}_2, \ldots, \mathrm{K}_m \rangle \qquad\qquad\qquad\quad \text{otherwise.} \end{cases} \tag{7}$$

III. If the i-th triple of the sequence is in the form of $s_i = \langle \omega'; q'; \emptyset \rangle$, then s_i is the last element of the sequence. The valuation ω' is the result of the successful computation. The valuation ω' is obtained with the probability q'.

3.4 Algebraic semantics of L_{prob}

Let $\langle \Im, \rho, p \rangle$ be a structure for L_{prob} , K be a program and W be the set of all valuations of variables from the program K. The measure

$$\mu : 2^W \to [0,1] \tag{8}$$

will be called probability distribution on W. Let us consider the set S of all such measures μ on W. Program K is interpreted as a partial function $\mathrm{K}_{\langle \Im, \rho, p \rangle} : S \to S$. If μ is an input distribution, such that $\mu(\omega_i) = \mu_i, i = 1, \ldots, l$, then $\mu' = \mathrm{K}_{\langle \Im, \rho, p \rangle}(\mu)$ is the output distribution such that $\mu'(\omega_i) = \mu', i = 1, \ldots, l$ (Fig. 1).

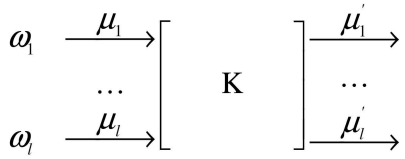

Fig. 1. Probabilistic interpretation of a program

Theorem 1. *[1]. Let $\langle \Im, \rho, p \rangle$ be a structure for L_{prob} with universe $U = \{u_1, u_2, \ldots, u_r\}$. For every program $\mathrm{K}(x_1, \ldots, x_1)$ interpreted in $\langle \Im, \rho, p \rangle$, we can construct, in an effective way, a matrix $K = [k_{ij}]_{i,j=1,\ldots,l}$, where $l = r^n$, such that for every input probability distribution μ, the output distribution μ' satisfies:*

$$\mu' = \mu \cdot K. \tag{9}$$

An element k_{ij} of matrix K corresponds to the probability that ω_j is the output valuation after computation of program K, provided that the computation starts at the valuation ω_i. The way of construction of the transition matrix K corresponding to program K is described in details in [1,5].
The following lemma enables us to determine, which positions k_{ij} of the matrix K are equal 0.

Lemma 1. *[1]. Let* K *be a program of the form while γ do* M *and let* M^{l+1} $(l = r^n)$ *denote the program*

$$\textit{begin}\ \underbrace{\textit{if}\ \gamma\ \textit{then}\ \mathrm{M}; \ldots;\ \textit{if}\ \gamma\ \textit{then}\ \mathrm{M}}_{l+1\ \textit{times}}\ \textit{end}.$$

Then for each distribution μ*,*

$$[\textit{while}\ \gamma\ \textit{do}\ \mathrm{M}]_{\langle \Im, \rho, p\rangle}(\mu) = [\textit{while}\ \gamma\ \textit{do}\ \mathrm{M}^{l+1}]_{\langle \Im, \rho, p\rangle}(\mu). \tag{10}$$

Furthermore,

$$k_{ij} = 0\ \textit{iff}\ m_{ij}^{l+1} = 0,\ i, j = 1, \ldots, l, \tag{11}$$

where k_{ij} *denote elements of the matrix* K *corresponding to the program* K *in the structure* $\langle \Im, \rho, p\rangle$*, and* m_{ij}^{l+1} *denote the elements of the matrix* M^{l+1} *corresponding to the program* M^{l+1}*.*

Moreover, by that lemma we can detect the places in program K, where it loops.

4. Non-deterministic Algorithmic Logic

In this subsection we briefly present a non-deterministic algorithmic language L_{ndtm}, being an extension of L_{alg} $(L \subset L_{alg} \subset L_{ndtm})$. The interpretation of the language L_{ndtm} will be understood as the pair $\langle \Im, U_0\rangle$, where $\Im$ is a structure for first-order language L with a finite universe U and U_0 denotes a subset of U. The non-deterministic interpretations $\langle \Im, U_0\rangle$ will be called *probe* if and only if $U_0 = U$.

4.1 Syntax of L_{ndtm}

We are given a fixed alphabet in which V is a set of individual and propositional variables, Ψ is a set of predicates, and Φ is a set of functors. Let F_o be a set of open formulas and T be set of terms. The set of non-deterministic iterative programs is defined in the same way, as for language L_{alg}, with nondeterministic assignment $x :=?$ and non-deterministic program construction *either* M_1 *or* M_2. The set of all formulas we denote by F_{ndtm}. It contains the classical first-order formulas constructed by means of the connectives $\wedge, \vee, \neg, \Rightarrow$ and quantifiers $\exists, \forall$. It also contains every expression α in the following form:

$$(\exists x)\beta(x), (\forall x)\beta(x), \tag{12}$$

$$(\beta \vee \lambda), (\beta \wedge \lambda), (\beta \Rightarrow \lambda), \neg\beta, \tag{13}$$

$$\Box \mathrm{K}\beta, \Diamond \mathrm{K}\beta, \tag{14}$$

where x is an individual variable, β, λ are arbitrary formulas and K is an arbitrary non-deterministic program.

The informal meaning of the formula $\Box K\beta$ is: *it is necessary that after realization* K *the formula* β *holds*. The informal meaning of the formula $\Diamond K\beta$ is: *it is possible that after realization* K *the formula* β *holds*, analogously to [8].
The fact, that the valuation ω in the structure $\langle \Im, U_0 \rangle$ satisfies the formula α will be denoted by $\langle \Im, U_0 \rangle, \omega \models \alpha$ (or simply $\omega \models \alpha$ when no confusion can arise).

The interpretation of open formulas and terms in the language L_{ndtm} is classical and this can be found in [8].
The element $\tau_{\langle \Im, U_0 \rangle}(\omega)$ of U is called *the value of the term* τ *in the structure* $\langle \Im, U_0 \rangle$ *at the valuation* ω . It will be shortly denoted by $\tau(\omega)$ when no confusion can arise.

4.2 Computational semantics of L_{ndtm}

By a *configuration* in structure $\langle \Im, U_0 \rangle$ we shall understand an ordered pair $\langle \omega; \overrightarrow{K} \rangle$, where ω is a valuation, and $\overrightarrow{K} = \{K_1, \ldots, K_n\}$ is a finite sequence of programs (which may be empty).

By a *tree of possible computations* of a program $K(x_1, x_2, \ldots, x_n)$ in the structure $\langle \Im, U_0 \rangle$ with a fixed input valuation ω we mean a tree $Tree(K, \omega, \Im, U_o)$ (shortly *Tree*), such that the configuration $\langle \omega; \overrightarrow{K} \rangle$ is the root of the *Tree*, *Rest* denotes a sequence of programs (it may be empty) and:

i. If a configuration $\langle \omega'$; *if* γ *then* M *else* N, *Rest*$\rangle$ is a vertex of Tree, then the unique son of this vertex is $\langle \omega'$; M, *Rest*$\rangle$ if $\omega' \models \gamma$ or $\langle \omega'$; N, *Rest*$\rangle$ otherwise.
ii. If a configuration $\langle \omega'$; *begin* M; N *end*, *Rest*$\rangle$ is a vertex of Tree, then the unique son of this vertex is $\langle \omega'$; M, N, *Rest*$\rangle$.
iii. If a configuration $\langle \omega'$; *while* γ *do* M, *Rest*$\rangle$ is a vertex of Tree, then the unique son of this vertex is $\langle \omega'$; M,*while* γ *do* M, *Rest*$\rangle$ if $\omega' \models \alpha$ or $\langle \omega'$;*Rest*$\rangle$ otherwise.
iv. If a configuration $\langle \omega'$;*either* M *or* N,*Rest*$\rangle$ is a vertex of Tree, then the left son of this vertex is $\langle \omega'$; M, *Rest*$\rangle$ and the right son is $\langle \omega'$; N, *Rest*$\rangle$.
v. If a configuration $\langle \omega'; x := \tau, Rest \rangle$ is a vertex of Tree, then the unique son of this vertex is $\langle \omega''; Rest \rangle$, where

$$\omega''(y) = \begin{cases} \tau(\omega') \text{ iff } y = x, \\ \omega'(y) \text{ iff } y \neq x. \end{cases} \tag{15}$$

vi. If a configuration $\langle \omega'; x :=?, Rest \rangle$ is a vertex of Tree, then the sons of this vertex are in the form $\langle \omega''; Rest \rangle$, where

$$\omega''(y) = \begin{cases} u \quad \text{iff } y = x \wedge u \in U_0, \\ \omega'(y) \text{ iff } y \neq x. \end{cases} \tag{16}$$

vii. If a configuration $\langle \omega'; \emptyset \rangle$ is a vertex of Tree, then it is a leaf of Tree, i.e. it has no sons.

Every path of the $Tree(\mathrm{K}, \omega, \Im, U_o)$ we called a *computation* of a program K in the structure $\langle \Im, U_0 \rangle$ at the input valuation ω. If $\langle \omega'; \emptyset \rangle$ is the leaf of the $Tree(\mathrm{K}, \omega, \Im, U_o)$, then the valuation ω' is called the *result* of the corresponding computation.

By the *interpretation* of program K in the structure $\langle \Im, U_0 \rangle$ we shall understand binary relation $\mathrm{K}_{\langle \Im, U_0 \rangle}$ in the set W of all valuations, such that $(\omega, \omega') \in \mathrm{K}_{\langle \Im, U_0 \rangle}$ iff ω' is a result of a computation of program K from the valuation ω in the structure $\langle \Im, U_0 \rangle$. The set of all results of the program K at the valuation ω in the structure $\langle \Im, U_0 \rangle$ we denote by

$$\mathrm{K}_{\langle \Im, U_0 \rangle}(\omega) = \{\omega' : (\omega, \omega') \in \mathrm{K}_{\langle \Im, U_0 \rangle}\} \tag{17}$$

The definition of interpretation of program K for different programs is as follows:

i. If K is of the form $x := \tau$, then

$$\mathrm{K}_{\langle \Im, U_0 \rangle} = \left\{(\omega, \omega') : \omega'(x) = \tau_{\langle \Im, U_0 \rangle}(\omega) \wedge \forall_{y \in V \setminus \{x\}} \omega'(y) = \omega(y)\right\}. \tag{18}$$

ii. If K is of the form $x :=?$, then

$$\mathrm{K}_{\langle \Im, U_0 \rangle} = \left\{(\omega, \omega^{'}) : \omega'(x) \in U_0 \wedge \forall_{y \in V \setminus \{x\}} \omega'(y) = \omega(y)\right\}. \tag{19}$$

iii. If K is of the form *while* $\neg\gamma$ *do* $x := x$, then the interpretation of program K will be denoted by I_γ and

$$I_\gamma = \{(\omega, \omega) : \omega \models \gamma\}. \tag{20}$$

iv. If K is of the form *begin* M; N *end*, and $M_{\langle \Im, U_0 \rangle}$, $N_{\langle \Im, U_0 \rangle}$ are the interpretations of the subprograms M, N, then

$$K_{\langle \Im, U_0 \rangle} = M_{\langle \Im, U_0 \rangle} \circ N_{\langle \Im, U_0 \rangle}. \tag{21}$$

v. If K is of the form *if* γ *then* M *else* N and $M_{\langle \Im, U_0 \rangle}, N_{\langle \Im, U_0 \rangle}$ are the interpretations of the subprograms M, N respectively, then

$$K_{\langle \Im, U_0 \rangle} = I_\gamma \cap M_{\langle \Im, U_0 \rangle} \cup I_{\neg\gamma} \cap N_{\langle \Im, U_0 \rangle}. \tag{22}$$

vi. If K is of the form *either* M *or* N , and $M_{\langle \Im, U_0 \rangle}, N_{\langle \Im, U_0 \rangle}$ are the interpretations of the subprograms M, N respectively, then

$$K_{\langle \Im, U_0 \rangle} = M_{\langle \Im, U_0 \rangle} \cup N_{\langle \Im, U_0 \rangle}. \tag{23}$$

vii. If K is of the form *while* γ *do* M, then

$$K_{\langle\Im,U_0\rangle} = \bigcup_{i=0}^{\infty} \left(I_\gamma \cap M_{\langle\Im,U_0\rangle} \cup I_{\neg\gamma}\right)^i \circ I_{\neg\gamma}. \tag{24}$$

In the above $\cup$, $\cap$ denote standard operations on sets and $\circ$ denotes the superposition of relations.

For an arbitrary valuation ω in the structure $\langle\Im, U_0\rangle$ we assume:

$$\langle\Im,U_0\rangle,\omega \models \Diamond \mathrm{K}\beta \text{ iff } \exists_{\omega' \in K_{\langle\Im,U_0\rangle}(\omega)} \langle\Im,U_0\rangle,\omega' \models \beta, \tag{25}$$

$$\langle\Im,U_0\rangle,\omega \models \Box \mathrm{K}\beta \text{ iff } \forall_{\omega' \in K_{\langle\Im,U_0\rangle}(\omega)} \langle\Im,U_0\rangle,\omega' \models \beta \text{ and}$$
$$\text{all computations of the program K at the valuation } \omega \text{ are finite.} \tag{26}$$

Remark 1. The formula $\Box \mathrm{K}\mathbf{1}$, where $\mathbf{1}$ is any formula of the form $\gamma \vee \neg\gamma$ will be called the *stopping formula* of the program K and it will be denoted by STOP(K). For this formula we have:

$$\langle\Im,U_0\rangle \models \text{STOP(K) iff all computations of the program K are finite.} \tag{27}$$

5. Results

Example 1. Let the probabilistic distribution on the set $U = \{1,2,3\}$ be fixed as:

$$\rho(1) = 1/2, \rho(2) = 1/3, \rho(3) = 1/6.$$

Let us consider an extremely simple program K interpreted in the universe U:

Algorithm 1 Program K

```
1: while x ≠ 3 do
2:     if x = 1 then
3:         x :=?
4:     else
5:         x := 1
6:     end if
7: end while
```

It is possible that, the sequence of states of program K looks as follows:

$$\omega(x) = 1, \omega(x) = 2, \omega(x) = 1, \omega(x) = 2, \omega(x) = 1, \omega(x) = 2, ... \tag{28}$$

However, if we consider the program K in the probabilistic structure $\langle \Im, \rho, p \rangle$ we have transition matrix for that program in the form:

$$K = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{29}$$

The probability that program K halts (when $x = 3$) is exactly 1 and the probability that the program K loops equals 0.

Now, lets consider program K in the non-deterministic structure $\langle \Im, U_0 \rangle$. It is easy to observe that in the tree $Tree(K, \omega, \Im, U_0)$ of possible computations of a program K contains infinite path. It means that, it is possible that program K loops. Thus the formula $\Box$ K$(x = 3)$, expressing fact that the all computations are finite, is not valid in the structure $\langle \Im, U_0 \rangle$.

Now, we can formulate our main results.

Definition 1. *The interpretations $\langle \Im, \rho, p \rangle$ and $\langle \Im, U_0 \rangle$ will be called* consistent *provided that, for each $i = 1, 2, \ldots, r$ the following holds:*

$$u \in U_0 \Leftrightarrow \rho(\mu) > 0. \tag{30}$$

Lemma 2. *Assume that the non-deterministic structure $\langle \Im, U_0 \rangle$ is consistent with the probabilistic structure $\langle \Im, \rho, p \rangle$. If*

$$\left\langle \omega^{(0)}; q^{(1)}; \overrightarrow{K}^{(1)} \right\rangle, \left\langle \omega^{(1)}; q^{(2)}; \overrightarrow{K}^{(2)} \right\rangle, \ldots, \left\langle \omega^{(i+1)}; q^{(i+1)}; \overrightarrow{K}^{(i+1)} \right\rangle, \ldots \tag{31}$$

is a probabilistic computation (finite or not) of the program K for the input valuation $\omega^{(0)}$, then

$$\left\langle \omega^{(0)}; \overrightarrow{K}^{(1)} \right\rangle, \left\langle \omega^{(1)}; \overrightarrow{K}^{(2)} \right\rangle, \ldots, \left\langle \omega^{(i+1)}; \overrightarrow{K}^{(i+1)} \right\rangle, \ldots \tag{32}$$

is a non-deterministic computation (finite or not) of the program K for the input valuation $\omega^{(0)}$.

Proof. The proof is by induction on i (with respect to the length of computation) and consists in analysis all possible cases of the program constructions. It is omitted for the sake of this paper compactness.

As an immediate consequence of Lemma 2 we have

Corollary 1. *Let* $\langle \Im, \rho, p \rangle$ *and* $\langle \Im, U_0 \rangle$ *be probabilistic and non-deterministic interpretation structures for* L_{alg} *respectively and* ω, ω' *be valuations. Let* μ *be a probability distribution such that* $\mu(\omega) > 0$*. From the above we check at once that in the case, when the output distribution* $\mu' = K_{\langle \Im, \rho, p \rangle}(\mu)$ *satisfies* $\mu'(\omega') > 0$*, exists a non-deterministic computation of program K starting with* ω *and ending at* ω'*.*

We can also prove the following

Theorem 2. *Let* K *be a program in the algorithmic language* L_{alg}*. Assume that the non-deterministic structure* $\langle \Im, U_0 \rangle$ *is consistent with the probabilistic structure* $\langle \Im, \rho, p \rangle$*. For any distribution* μ *and any valuation* ω *in the structure* $\langle \Im, \rho, p \rangle$*, if* $\mu(\omega) = 1$ *and* $\langle \Im, \rho, p \rangle, \omega \models \mathrm{K}(P(\gamma) > 0)$*, then* $\langle \Im, U_0 \rangle, \omega \models \Diamond \mathrm{K}\gamma$*.*

Proof. Proof is omitted. It proceeds by induction on the length of the program computation.

Now, we will formulate the non-deterministic analogon of the Lemma 1.

Theorem 3. *Let* K *be a program in the algorithmic language* L_{alg}*. Assume that the non-deterministic structure* $\langle \Im, U_0 \rangle$ *is consistent with the probabilistic structure* $\langle \Im, \rho, p \rangle$*. If in the tree of all possible probabilistic computations of the program K, starting with distribution satisfying* $\mu(\omega) = 1$*, exists a path of length longer than l, where* $l = |W| = r^n$*, then* $\langle \Im, U_0 \rangle, \omega \models \neg$ STOP(K)*.*

5.1 Algebraic semantics of L_{ndtm}

In this approach we suggest using the calculus abstract over probabilities. More precisely, we resign from precise probability values by replacing them with terms possibility and necessity. We define formalism by analogy to algebraic semantics of iterative probabilistic algorithms and using in matrices only two values: 0 and e, expressing impossibility and possibility, respectively.

Definition 2. *Let* $\langle \{0, e\}, +, \cdot = \rangle$ *be the algebraic structure. The operations* $\cdot, +$ *are defined on the set* $\{0, e\}$ *in the following way.*

Table 1. Operation 1

$\cdot$	0	e
0	0	0
e	0	e

Table 2. Operation 2

+	0	e
0	0	e
e	e	e

The following theorem we will formulate without proof.

Theorem 4. *Let* $\langle \Im, U_0 \rangle$ *be a structure for* L_{ndtm} *with the universe* $U = \{u_1, u_2, \ldots, u_r\}$. *Let* $\mathrm{K}(x_1, \ldots, x_n)$ *be a program interpreted in the structure* $\langle \Im, U_0 \rangle$ *and* ω_i, ω_j *be any valuations from the set* $W = \{\omega_1, \ldots, \omega_l\}, l = r^n$. *For every program* $\mathrm{K}(x_1, \ldots, x_n)$, *we can construct a matrix*

$$E_K = [e_{ij}]_{i,j=1,\ldots,l}, \tag{33}$$

such that $e_{ij} \in \{0, e\}$ *and*

$$e_{ij} = \begin{cases} e \text{ iff } (\omega_i, \omega_j) \in K_{\langle \Im, U_0 \rangle} \\ 0 \text{ otherwise.} \end{cases} \tag{34}$$

Proof. The detailed proof is omitted. We only give the main ideas of construction of the transition matrix E_K corresponding to program K:

i. If K is of the form $x := \tau$, then matrix E_K is defined as:

$$e_{ij} = \begin{cases} e \text{ iff } \omega_j(x) = \tau_{\langle \Im, U_0 \rangle}(\omega_i) \wedge \forall_{y \in V \setminus \{x\}} \omega_j(y) = \omega_i(y), \\ 0 \text{ otherwise.} \end{cases} \tag{35}$$

ii. If K is of the form $x :=?$, then matrix E_K is defined as:

$$e_{ij} = \begin{cases} e \text{ iff } \omega_j(x) \in U_0 \wedge \forall_{y \in V \setminus \{x\}} \omega_j(y) = \omega_i(y) \\ 0 \text{ otherwise.} \end{cases} \tag{36}$$

iii. If K is of the form *while* $\neg\gamma$ *do* $x := x$, then matrix corresponding to program K will be denoted by I_γ and defined as:

$$e_{ij} = \begin{cases} \text{e iff } i = j \wedge \omega_i \models \gamma, \\ 0 \text{ otherwise.} \end{cases} \tag{37}$$

iv. If K is of the form *begin* M; N *end*, and the matrices E_M and E_N for the subprograms M, N are constructed, then

$$E_K = E_M \cdot E_N. \tag{38}$$

v. If K is of the form *if* γ *then* M *else* N, and the matrices E_M and E_N for the subprograms M, N are constructed, then

$$E_K = I_\gamma \cdot E_M + I_{\neg\gamma} \cdot E_N. \tag{39}$$

vi. If K is of the form *either* M *or* N, and the matrices E_M and E_N for the subprograms M, N are constructed, then

$$E_K = E_M + E_N. \tag{40}$$

vii. If K is of the form *while* γ *do* M, then the matrix E_K is defined as:

$$E_K = \lim_{i\to\infty} \sum_{j=0}^{i} \left(I_\gamma \cdot E_M + I_{\neg\gamma}\right)^i \cdot I_{\neg\gamma}. \tag{41}$$

The reader can easily verify that the limit in the equation 41 always exists.

Example 2. Let us reconsider program K from Example 1. The transition matrix E_K corresponding to the program K, determined in accordance with the above rules, is as follows:

$$E_K = \begin{bmatrix} 0 & 0 & e \\ 0 & 0 & e \\ 0 & 0 & e \end{bmatrix}. \tag{42}$$

6. Final remarks

The formalism proposed in subsection 5.1 is useful in situations in which we interest if it is possible that program K halts and we do not have to know the accurate values of probabilities, except the fact that they are positive (cf. [1]).

Theorem 2 and Theorem 3 point at a direction for further work. The main goal of our research is to explain the dependencies between non-deterministic and probabilistic interpretation of the iterative programs. It will be the subject of next paper.

References

[1] Dańko W.: The Set of Probabilistic Algorithmic Formulas Valid in a Finite Structure is Decidable with Respect to Its Diagram, Fundamenta Informaticae 19, 1993, pp. 417–431.

[2] Feldman Y. A., Harel D.: A Probabilistic Dynamic Logic, Journal of Computer and System Sciences 28, 1984, pp. 193–215.

[3] Hare D., Pratt V. R.: Nondeterminism in Logics of Programs, Proc. of the 5th Symp. on Principles of Programming Languages SIGACT - SIGPLAN, ACM, 1978, pp. 23–25.
[4] Harel D., Kozen D., Tiuryn J.: Dynamic Logic, MIT Press, MA, 2000.
[5] Koszelew J.: The Methods for Verification Properties of Probabilistic Programs, Ph.D. thesis, Inst. of Comput. Sci., Polish Academy of Sciences, 2000.
[6] Kozen D.: Semantics of probabilistic programs, Journal of Computer and System Sciences 22, 1981, pp. 328–350.
[7] Kozen D.: A Probabilistic PDL, Journal of Computer and System Sciences 30(2), 1985, pp. 162–178.
[8] Mirkowska G., Salwicki A.: Algorithmic Logic, D. Reidel Publ. Co. & PWN, 1987.

PROBABILISTYCZNE I NIEDETERMINISTYCZNE SEMANTYKI DLA PROGRAMÓW ITERACYJNYCH

Streszczenie W pracy rozważane są, na gruncie logiki programów, probabilistyczne i niedeterministyczne interpretacje programów iteracyjnych. Uwaga autorów skupia się na związkach miedzy tymi interpretacjami. Punktem wyjścia są formalne definicje semantyk dla obu podejść. Główny nacisk został położony na wyrażalność własności stopu w tych semantykach.

Słowa kluczowe: Słowa kluczowe: program niedeterministyczny, program probabilistyczny, obliczenie probabilistyczne, obliczenie niedeterministyczne

Artykuł zrealizowano w ramach pracy badawczej S/WI/3/2008.