Nondeterministic programming within the frame of first order classical logic, Part 2

By T. GERGELY and L. ÚRY

In this part we develop a mathematical theory of nondeterministic sequential programming by using the mathematical tools introduced in the first part (see GERGELY and ÚRY (1980)). Our investigation is concentrated around the problem of completeness. While this we introduce an appropriate complete descriptive language and a complete calculus in the spirit of Floyd and Hoare. The usage of the calculus is illustrated by three examples.

4. Definable games

We aim at developing a theory of non-deterministic programming within the frame of first order language and therefore we have to formalize the games providing consideration of nondeterminism. Unfortunately the basic notions belonging to games introduced in Section 2 are not that of first order language. To keep ourselves within the frame of first order logic we have to consider such version of these notions that are parametrically definable. Thus we permit only parametrically definable games, strategies etc. However it might happen that the property "to be a definable game" is not definable while each game is definable. By using the arithmetization of formal languages we show below that this is not the case.

Let Ax^* be a ϑ-type system and let $\mathfrak{A}\in Md(Ax^*)$ be arbitrary. There are several definable bijections between $A\times A$ and A. Let us fix one of them and denote it by *pair*. Let *left* and *right* be the two components of the inverse of the function *pair*, i.e. for any $x, y\in A$

$$left\,(pair\,(x, y))=x \text{ and } right\,(pair\,(x, y))=y.$$

Remember that in any model $\mathfrak{A}\in Md(Ax^*)$ induction can be done by inner time $T_{\mathfrak{A}}$ i.e. by the set $\{t\in A\,|\,\mathfrak{A}\models\zeta^*[t]\}$ (see Definition 3.10). Thus the usual notion of sequence (see Section 1.7) has to be modified in the following way.

Definition 4.1. Let $\mathfrak{A}\in Md(Ax^*)$ and let D be an initial segment of inner time $T_{\mathfrak{A}}$. A function $s\colon D\to A$ is a *finite* (in $\mathfrak{A}$) sequence. If $D=[0, n-1]$ then s is called an *n-long sequence*. A function $s\colon T_{\mathfrak{A}}\to A$ is said to be a *sequence* or a *ζ-long sequence*. □

Let $K \subset A \setminus \{0\}$ be a definable set and let ${}^*K$ denote the set of all parametrically definable finite (in $\mathfrak{A}$) sequences containing only elements of K. Let Λ denote the empty sequence.

Lemma 4.2. "To be an element of ${}^*K$" is definable in $\mathfrak{A}$.

Proof. Let φ be the formula which defines K in $\mathfrak{A}$, i.e. $\operatorname{Var}\varphi = \{x\}$ and $\mathfrak{A} \models \varphi[a] \Leftrightarrow a \in K$.

Let us consider the following formula:

$$\varphi^*(s) \overset{d}{=} \exists t\big(left(s) \leqq t \wedge \zeta(t) \wedge \forall i(i \leqq left(s) \to \varphi[\Gamma(right(s), left(s), i)/x])\big),$$

where Γ is the well-know Gödel-function (see e.g. in MENDELSON (1964)). Since the functions Γ, *left* and *right* are definable in Ax^* the formula φ^* is equivalent to a formula of F^V_{3*}. For the sake of convenience we suppose that a formula φ^* (e.g. $i \leqq left(a)$) at the same time denotes the corresponding formula of F^V_{3*}, i.e. $\varphi^* \in F^V_{3*}$.

Now let $s \in A$ be an element such that

$$\mathfrak{A} \models \varphi^*[s].$$

Let us take the following sequence:

$$f_s(i) \overset{d}{=} \Gamma(right(s), left(s), i) \quad \text{for any} \quad i \leqq left(s).$$

Now we prove that $s \mapsto f_s$ is a surjective map, i.e. the elements of ${}^*K$ are coded by s but these codes are not unique.

Let $f\colon [0, n] \to K$ be a parametrically definable function in $\mathfrak{A}$. By using the generalized Sequence Number Theorem (see GERGELY and ÚRY (1978), Theorem 2.8) there is a $b \in A$ such that for any $i \in [0, n]$, $f(i) = \Gamma(b, n, i)$.

Let $s \overset{d}{=} pair(c, b)$. If f is finite in $\mathfrak{A}$ then by using the fact that Ax^* is a system and the definition of Γ we have $\mathfrak{A} \models \zeta^*[s]$ i.e. there is a $t \in A$ such that $s \leqq t$ and $\mathfrak{A} \models \zeta[s]$. Thus $\mathfrak{A} \models \varphi^*[s]$ and s codes the given function f. □

Lemma 4.3. The following functions are parametrically definable in $\mathfrak{A}$ (and also in Ax^*)

$$pair\colon {}^*K \times K \to {}^*K,$$

$$left\colon {}^*K \to {}^*K,$$

$$right\colon {}^*K \to A. \quad \square$$

Note that here the functions *pair*, *left* and *right* are defined on the sequences. However we use the same notation as on page 355 because the present case can be obtained by iterative usage of the functions introduced there. Thus this notion does not lead to ambiguity. For the sake of convenience we suppose that the empty sequence Λ is coded by 0.

Further on we do not distinguish the elements of ${}^*K$ from the corresponding elements of A coding them.

Definition 4.4.

$$length(a) = \begin{cases} 0 & \text{if} \quad a = \Lambda \\ length(left(a)) + 1 & \text{otherwise} \end{cases}$$

Definition 4.5. A set $V \subset^*(A \setminus \{0\})$ is called a tree iff
(i) $\Lambda \in V$,
(ii) $v \in V \Rightarrow left(v) \in V$.

For any V and $v \in V$ take $S_V(v) \stackrel{d}{=} \{w \in V \mid left(w) = v\}$.

A tree V is called a *path* iff *left*: $V \setminus \{\Lambda\} \to V$ is an injection, i.e. any $v \in V$ has at most one successor.

A tree V is *definable in* $\mathfrak{A}$ iff it is parametrically definable in $\mathfrak{A}$ as a unary relation on A. □

Note that because 0 codes Λ it is sufficient to use $A \setminus \{0\}$ instead of A in the above definition. Thus 0 can be maintained to denote the end of a sequence.

Definition 4.6. A *trace in* $\mathfrak{A}$ is a function of the form f: $V \to {}^nA$ (for some natural number n) where V is a tree in $\mathfrak{A}$. If f is parametrically definable then it said to be a *definable trace in* $\mathfrak{A}$. □

Definition 4.7. Let V be a trace in $\mathfrak{A}$. A *game-frame* in $\mathfrak{A}$ is a pair $GF=(V, C)$ where $C \subset V$ is an arbitrary set. C is called the nodes of choice of V. A run in a *game-frame* $GF=(V, C)$ is a subtree $R \subset V$ such that

(i) if $c \in C \cap R$ then c has a unique successor in R,
(ii) if $r \in R \setminus C$ then $S_V(r) = S_R(r)$.

A *strategy in a game-frame* G is a function *str*: $C \to \mathrm{rg}\, str$ in such a way that for any $c \in C$, $str(c) \in S_V(c)$.

A game-frame is *definable* in $\mathfrak{A}$ iff V and C are parametrically definable. A run R of GF is *definable* in $\mathfrak{A}$ if both GF and R are parametrically definable in $\mathfrak{A}$.

A strategy *str* in GF is *definable* in $\mathfrak{A}$ iff GF itself and the function *str* are parametrically definable. □

Definition 4.8. Let $GF=(V, C)$ be an arbitrary game-frame and let *str* be a strategy in GF. A run R is said to be generated by the strategy *str* if $\{str(c)\} = S_R(c)$ for any $c \in C \cap R$. □

We note that if *str* is definable then there exists a minimal definable run generated by *str* and this is denoted by R_{str}.

Definition 4.9. A tree V in $\mathfrak{A}$ is *finitary* iff for any $v \in V$ there is a $d \in A$ such that
1) $\mathfrak{A} \models \zeta[d]$,
2) if *pair* $(v, e) \in S_V(v)$ then $e \leqq d$, i.e. any node of V has only finitely many successors (in $\mathfrak{A}$). □

Definition 4.10. A *game* in $\mathfrak{A}$ is a quadruple $G=(V, C, \Gamma_A, \Gamma_B)$ where (V, C) is a game-frame and Γ_A, Γ_B are disjoint sets of paths in V.

A game $(V, C, \Gamma_A, \Gamma_B)$ is definable in $\mathfrak{A}$ iff
(i) (V, C) is a definable game-frame,
(ii) Γ_A and Γ_B are definable sets,
(iii) each path of Γ_A and Γ_B is definable. □

Definition 4.11. A strategy *str* in the game-frame $GF=(V, C)$ is a winning (non-losing) strategy of player A in the game $(V, C, \Gamma_A, \Gamma_B)$ iff each definable path of R_{str} belongs to Γ_A (not to Γ_B). □

According to this definition a run R of the game is non-losing for player A if none of its paths belong to any of Γ_B and R.

Thus a stategy of the player A is a winning (non-losing) one in the game $(V, C, \Gamma_A, \Gamma_B)$ iff moving accordingly the course of game realizes only paths belonging to Γ_A (not belonging to Γ_B). If we are interested only in non-losing strategies then it is enough to consider games of the form $(V, C, \emptyset, \Gamma_B)$.

In order to show that among others the property "to be a definable tree" is expressible by a first order formula we need the following well known theorem about the existence of a universal formula, though the precise form of this formula is not necessary to our investigation.

Theorem 4.12. (On universal formula.) Let us fix an arbitrary ϑ-type system Ax^*. There is a recursive map $F^V_{\vartheta *} \to \mathbf{N}$ $(\varphi \mapsto \ulcorner\varphi\urcorner$ where $\ulcorner\varphi\urcorner$ denotes the Gödel number of the formula φ) and a formula $Valid\ (g, \vec{a}, x) \in F^V_{\vartheta *}$ such that for any $\mathfrak{A} \in Md(Ax^*)$ and $\varphi \in F^V_{\vartheta *}$, $\mathfrak{A} \models \varphi[\vec{a}, x]$ iff $\mathfrak{A} \models Valid[\ulcorner\varphi\urcorner, \vec{a}, x]$.

Proof. See in MENDELSON (1964).

Theorem 4.13. (Expressibility.) For any Ax^* the following properties are expressible in Ax^* by using the formulas of $F^V_{\vartheta *}$:

a) "to be a definable tree",
b) "to be a definable path in a definable tree",
c) "to be a definable game-frame",
d) "to be a definable run in a definable game-frame",
e) "to be a definable strategy in a definable game-frame",
f) "to be a non-losing definable run of a definable game",
g) "to be a non-losing definable strategy of a definable game",
h) "there is a non-losing definable strategy in a definable game".

Proof. Each statement can be proved by the same method. Thus we detail only the proof of property a) by showing the existence of the formula corresponding to this case.

a) Let Ω be the variable symbol the values of which correspond to the Gödel number of the formula φ that parametrically defines a tree in $\mathfrak{A}$ and let $\vec{a}$ be the vector of variable symbols the values of which correspond to the parameters.

The property "to be an element of tree V" can be defined by using the formula *Valid* (see Theorem 4.13). Namely *Valid* $(\Omega, \vec{a}, x)$ means that the element x is an element of the tree Ω with parameter $\vec{a}$ (here of course we identify the definable objects with the Gödel number of the appropriate formulas defining them). Having this defining formula we can construct the formula defining the property a). Namely

$$Tree(\Omega, \vec{a}) \overset{d}{=} Valid(\Omega, \vec{a}, \Lambda) \wedge \forall x\,(Valid(\Omega, \vec{a}, x) \to Valid(\Omega, \vec{a}, left(x))).$$

Let $\mathfrak{A} \in Md(Ax^*)$. Now for fixed $\underset{\sim}{\Omega}, \underset{\sim}{\vec{a}} \in A$ take

$$V_{\underset{\sim}{\Omega}\underset{\sim}{\vec{a}}} \overset{d}{=} \{v \in A \mid \mathfrak{A} \models Valid[\underset{\sim}{\Omega}, \underset{\sim}{\vec{a}}, v]\}.$$

It is clear that $V_{\underset{\sim}{\Omega}, \underset{\sim}{\vec{a}}}$ is a parametrically definable tree in $\mathfrak{A}$. Moreover if φ defines a tree V by parameters $\vec{a}$ then, using 4.12, we have $V = V_{\ulcorner\varphi\urcorner, \vec{a}}$.

b) Let $Path(\Omega, g, \vec{a}, \vec{b}) \overset{d}{=} \forall x\,(Valid(g, \vec{b}, x) \to Valid(\Omega, \vec{a}, x)) \wedge$

$\forall x\,\forall y\,\{Valid(g, \vec{b}, x) \wedge Valid(g, \vec{b}, y) \wedge left(x) = left(y) \to x = y\}.$

It is clear that if $\mathfrak{A} \in Md(Ax^*)$ and $\underset{\sim}{\Omega}, \underset{\sim}{g}, \underset{\sim}{a}, \underset{\sim}{\vec{b}} \in A$ then $V_{\underset{\sim}{g}, \underset{\sim}{\vec{b}}}$ is a definable path in $V_{\underset{\sim}{\Omega}, \underset{\sim}{\vec{a}}}$. We omit the proof of properties c), e) and f).

d) Let $GF(\Omega_V, \Omega_C, \vec{a})$ be the formula corresponding to property c)

$$R(\Omega_V, \Omega_C, \vec{a}, r, \vec{b}) \overset{d}{=} GF(\Omega_V, \Omega_C, \vec{a}) \wedge Tree(r, \vec{b}) \wedge \forall x[Valid(r, \vec{b}, x) \to$$

$$Valid(\Omega_V, \vec{a}, x)] \wedge \forall x \forall y[Valid(r, \vec{b}, x) \wedge Valid(\Omega_V, \vec{b}, y) \wedge left(y) = x \wedge$$

$$\neg Valid(\Omega_C, \vec{a}, x) \to Valid(r, \vec{b}, y)] \wedge$$

$$\forall x\,\forall y\,\forall z\,[Valid(r, \vec{b}, x) \wedge Valid(r, \vec{b}, y) \wedge Valid(r, \vec{b}, z) \wedge$$

$$left(y) = x \wedge left(z) = y \wedge Valid(\Omega_C, \vec{a}, x) \to y = z].$$

It is evident that this formula is good.

g) Let

$$NL(\Omega_V, \Omega_C, \Omega_{\Gamma_B}, \vec{a}) \overset{d}{=} \exists r\,\exists \vec{b}\,(R(\Omega_V, \Omega_C, \vec{a}, r, \vec{b}) \wedge$$

$$\forall g\,\forall \vec{c}\,(Path(r, g, \vec{b}, \vec{c}) \to \neg Valid(\Omega_{\Gamma_B}, pair(\vec{a}, \vec{c}), g))).$$

This formula means that there is a run r with parameters $\vec{b}$ such that r is a run of the game defined by Ω_V and Ω_C and no path g with parameter $\vec{c}$ of r belongs to Ω_{Γ_B}. □

Now we reformalize the well-know König's lemma for the case of definable finitary trees.

Lemma 4.14. (König.) Let V be a parametrically definable and finitary tree in $\mathfrak{A}$ such that for any $d \in A$ satisfying $\mathfrak{A} \models \zeta^*[d]$ there is a d-long definable path. Then there is a ζ-long definable path as well.

Proof. Any proof of the original form of this lemma can be repeated because the definability of the tree and paths provides the expressibility of each step of the proof in the first order language. Details are omitted. □

5. Nondeterministic programming language

We introduce a nondeterministic programming language of the autocode level. This level provides a relatively simple definition of semantics considering time conditions as well. This description is done by using games introduced in Section 4. Having exact semantics we turn to the investigation of the question of descriptive languages, which is one of the foundamental component of a mathematically based programming theory.

Now we introduce a ϑ-type nondeterministic programming language NP_ϑ. The programs of the language NP_ϑ might be "executed" in the models of on arbitrary ϑ-type system. Altogether the "meaning" of the programs varies by systems and by their models.

Let us fix a denumerable infinite set Y of variable symbols.

Definition 5.1. The set U_ϑ of ϑ-type nondeterministic commands consists of the following elements:

(i) j: $y \leftarrow \tau$,

(ii) j: $\square y \leqq \tau$,

(iii) j: if χ then $\square$ $k_1, \ldots, k_r$

where $\square \in \{any, every\}$, $j, k_1, \ldots, k_r$ are natural numbers, $y \in Y$, $\tau \in T_\vartheta^Y$ and $\chi \in Q_\vartheta^Y$. □

Further on sign $\square$ in commands stands either for *any* or *every*.

Thus the commands of U_ϑ have the form j: u, where the natural number j is said to be the *label* of the command.

Definition 5.2. A ϑ-type nondeterministic program p is a nonempty sequence of ϑ-type commands $p = \langle i_0: u_0, \ldots, i_n: u_n \rangle \in U_\vartheta^+$ in which no two commands have the same labels, i.e. for any $j, k \in [0, n]$ if $j \neq k$ then $i_j \neq i_k$.

Let NP_ϑ denote the set of all ϑ-type nondeterministic programs:

$NP_\vartheta \overset{d}{=} \{p \mid p$ is a ϑ-type nondeterministic program$\}$. □

Note that we often write a program p in the form of column

$$\begin{array}{l} i_0 : u_0 \\ \ldots \\ i_n : u_n \end{array}$$

instead of the form of row. In the column form we always assume that $i_0 < \ldots < i_n$.

For any nondeterministic program $p = \langle i_0: u_0, \ldots, i_n: u_n \rangle \in NP_\vartheta$ we use the following notations:

(i) $\operatorname{Var} p \overset{d}{=} \bigcup_{j \in [0,n]} \operatorname{Var}(i_j: u_j)$ denotes the set of variable symbols occuring in the program p. Here $\operatorname{Var}(i_j: u_j)$ is the set of variable symbols occuring in the command i_j: u_j defined as follows

$$\operatorname{Var}(j: y \leftarrow \tau) \overset{d}{=} \operatorname{Var} \tau \cup \{y\},$$

$$\operatorname{Var}(j: \square y \leqq \tau) \overset{d}{=} \operatorname{Var} \tau \cup \{y\},$$

$$\operatorname{Var}(j: \textit{if } \chi \textit{ then } \square\ k_1, \ldots, k_r) \overset{d}{=} \operatorname{Var} \chi.$$

For the sake of convenience we often use Y_p instead of $\operatorname{Var} p$.

(ii) $i_{n+1} \overset{d}{=} \min\{k \mid k \notin \{i_0, \ldots, i_n\}\}$.

The programming language NP_ϑ seems to be far too weak though it is powerful enough as shown in its deterministic counter part in GERGELY and ÚRY (1978). Now we note only the command j: *if* $x = x$ *then* $\square$ $k_1, \ldots, k_n$ is the same as j: *goto* $\square$ $k_1, \ldots, k_n$. We use the latter in the language NP_ϑ as an abbreviation.

Since the "meaning" of a program is its "execution" let us consider the case when the execution takes place in an arbitrary model of a given ϑ-type system Ax^*.

Definition 5.3. Let us fix a system Ax^*. Let us consider an arbitrary model $\mathfrak{A} \in Md(Ax^*)$ and an arbitrary program $p = \langle i_0 \colon u_0, \ldots, i_n \colon u_n \rangle \in NP_\vartheta$.

Let q: $\mathrm{Var}\, p \to A$ be an arbitrary evaluation of the variable symbols of p.

Let $G_q = (V_q, C_q)$ be a game-frame and $f_q = (l, s)$: $V_q \to \mathbf{N} \times A^{\mathrm{Var}\, p}$ be a trace with the following properties:

1. G_q and f_q are parametrically definable in $\mathfrak{A}$;
2. (i) $\mathfrak{A} \models \zeta^* [length(v)]$ for any $v \in V_q$,
(ii) $l(0) = i_0$ and $s(0) = q$,
(iii) if $l(v) \notin \{i_m \mid m \leqq n\}^\bullet$ then there exists no successor of v, i.e. $S_{V_q}(v) = \emptyset$;
3. Let us suppose that $l(v) = i_m$

(i) if $u_m = y \leftarrow \tau$ then $S_{V_q}(v) = \{pair(v, 0) \stackrel{\mathrm{d}}{=} w\}$, $l(w) = i_{m+1}$,

$$s(w)(x) = \begin{cases} s(v)(x) & \text{if } x \in \mathrm{Var}\, p \setminus \{y\}, \\ \tau_{\mathfrak{A}}[s(v)] & \text{if } x = y, \end{cases}$$

(ii) if $u_m = \square y \leqq \tau$ then $S_{V_q}(v) = \{pair(v, e) \mid 0 \leqq e \leqq \tau_{\mathfrak{A}}[s(v)]\}$ and for any $w \in S_{V_q}$, $l(w) = i_{m+1}$,

$$s(w)(x) = \begin{cases} s(v)(x) & \text{if } x \in \mathrm{Var}\, p \setminus \{y\}, \\ right(w) & \text{if } x = y, \end{cases}$$

(iii) if $u_m = $ *if* χ *then* $\square$ $k_1, \ldots, k_r$ then
a) if $\mathfrak{A} \models \neg \chi[s(v)]$ then $S_{V_q}(v) = \{pair(v, 0)\}$ and $l(w) = i_{m+1}$ for $w \in S_{V_q}(0)$,
b) if $\mathfrak{A} \models \chi[s(v)]$ then $S_{V_q}(v) = \{pair(v, k_j) \mid j \in [1, r]\}$ and $l(w) = right(w)$ for any $w \in S_{V_q}(v)$.
In both cases a) and b) for any $w \in S_{V_q}(v)$, $s(w) = s(v)$.
The set of nodes of choice in the cases (ii) and (iii) is defined as follows

$$C_q \stackrel{\mathrm{d}}{=} \{v \in V_q \mid l(v) = i_m \text{ and } u_m \text{ is either } any\ y \leqq \tau \text{ or } if\ \chi\ then\ any\ k_1, \ldots, k_r\}.$$

The game-frame of the above properties is said to be the *q-game-frame associated* to the program p; the trace f_q of above properties is said to be a *trace* of the program p in the model $\mathfrak{A}$ starting with input data q. □

Now we take the points of the definition one by one and show what conditions of programming are indicated by them. The q-game frame (V_q, C_q) describes the nondeterminism of the trace of a program; the function $f_q = (l, s)$ shows the actual value of the variable symbols of the program in each moment of execution.

If the program gets in a state represented by the node $v \in V_q$ then in the next step the command labelled by $l(v)$ will be executed with the $s(v)$ evaluation of the variable symbols.

Condition 1 is in accordance with the assumption that the first order language is used to describe the ϑ-type models.

Condition 2 (ii) provides that the game goes on for time $T = \zeta^*$ and it only stops when the situation 2 (iii) arises, i.e. when the control $l(v)$ gets such a label that does not occur among $i_0, \ldots, i_n$. So the termination of the trace is equivalent

to a jump onto such label. This is the reason why a special command *stop* is not included in U_ϑ.

According to 2 (ii) a trace starts always with the first command of the program, i.e. with the label i_0 with an evaluation q given in advance. This is the reason why a *start* command is not used.

Conditions of point 3 describe the one step transition of the program execution in the usual way. If a no control command with label i_m is executed then after the execution the control gets on label i_{m+1}. This is why the virtual label i_{n+1} was introduced to indicate the termination of the execution.

While executing commands with nondeterminism, i.e. with the sign $\square$, the game-tree branches out according to the possible choices. The cases *any* and *every* differ from one another simply whether player A or B moves. The correctness of the above definition is ensured by the following

Lemma 5.4. For an arbitrary program $p \in NP_\vartheta$, $\mathfrak{A} \in Md(Ax^*)$ and q: $\mathrm{Var}\, p \to A$ there is a unique trace of p in $\mathfrak{A}$ starting with q and there is a unique associated q-game-frame GF_q. Moreover, the parameters of the formulas that define the trace f_q and the game-frame GF_q are the values of q and these definitions are universal (i.e. in each model of Ax^* they are defined by the same formula).

Proof. The uniqueness means that the associated game-frame GF_q and the corresponding trace f_q are unique. Let GF'_q, and f'_q be another q-game-frame and another trace for which conditions 1—3 of Definition 5.3 hold. Let v be an element of V with minimal length such that $v \notin V'$ or $v \in V'$ but $f_q(v) \neq f'_q(v)$. This minimum exists because both (GF_q, f_q) and (GF'_q, f'_q) are parametrically definable. Since $v \neq \Lambda$ so $w = left\,(v) \in V$ and by the minimality of v we have $w \in V'$ and $f(w) = f'(w)$. If so then by using conditions 1—3 we have that $v \in V'$ also holds and $f(v) = f'(v)$ which is a contradiction. To prove the existence of the trace we construct a recursive equation, the solution of which gives a trace of p in $\mathfrak{A}$ starting with q.

$$\varrho(v, l, y_1, \ldots, y_k, a_1, \ldots, a_k) \leftrightarrow \left(l = i_0 \wedge v = \Lambda \wedge \bigwedge_{j=1}^{k} y_j = a_j\right)$$

$$\vee \exists v^*, l^*, y_1^*, \ldots, y_k^* \Big\{ \varrho(v^*, l^*, y_1^*, \ldots, y_k^*, a_1, \ldots, a_k) \wedge l^* \in \{i_m \mid m \leqq n\} \wedge \xi^*(length\, v) \wedge$$

$$\bigwedge_{\substack{m=0 \\ u_m = y_s \leftarrow \tau}}^{n} l^* = i_m \to \Big\{ v = (v^*, 0) \wedge \bigwedge_{\substack{i \neq s \\ i=1}}^{k} y_i^* = y_i \wedge y_s = \tau[\vec{y}^*/\vec{y}] \wedge l = i_{m+1} \Big\} \wedge$$

$$\bigwedge_{m=0}^{n} l^* = i_m \to \Big\{ v = (v^*, y_s) \wedge \bigwedge_{\substack{i \neq s \\ i=1}}^{k} y_i^* = y_i \wedge y_s = \tau[\vec{y}^*/\vec{y}] \wedge l = i_{m+1} \Big\} \wedge$$

$$\bigwedge_{\substack{m=0 \\ u_m = if\ \chi\ then\ \square k_1, \ldots, k_r}}^{n} l^* = i_m \to \Big[\bigwedge_{i=1}^{k} y_i^* = y_i \wedge (\neg\chi \to v = (v^*, 0) \wedge l = i_{m+1}) \wedge$$

$$(\chi \to \bigvee_{i=1}^{r} v = (v^*, k_i) \wedge l = k_i) \Big] \Big\}.$$

Applying Theorem 3.15 this equation has a minimal solution say $\varrho^* \in F_{\vartheta^*}$. It is eviden that

$$\Omega_V(w, a) \overset{d}{=} \exists l, y_1, \ldots, y_k \varrho^*(w, l, y_1, \ldots, y_k, a_1, \ldots, a_k)$$

defines a tree in $\mathfrak{A}$ and if

$$\Omega_C(w, \vec{a}) \overset{d}{=} \exists l, y_1, \ldots, y_k \left[\varrho^*(v, l, y_1, \ldots, y_k, a_1, \ldots, a_k) \wedge \bigvee_{\substack{m=0 \\ u_m = any\ y_s \\ u_m = if\ \chi\ then\ any\ k_1, \ldots, k_r}}^{n} l = i_m \right]$$

then (Ω_V, Ω_C) defines the associated q-game-frame GF_q and ϱ^* parametrically defines the trace of p in $\mathfrak{A}$ starting with q.

Note that since the solution of the above recursive equation is the same in any model of Ax^* thus ϱ^* defines the trace of p in any model of Ax^*. The universality of the definition of GF_q is evident from the construction. □

6. Descriptive language

Let Ax^* be a ϑ-type system and $\mathfrak{A} \in Md(Ax^*)$ an arbitrary model. Let $p = (i_0 \colon u_0, \ldots, i_n \colon u_n) \in NP_\vartheta$ and $q \colon \operatorname{Var} p \to A$ an arbitrary input. We emphasize that the input q is arbitrary but fixed. The q-game-frame associated with the program p is $GF_q = (V_q, C_q)$.

Let Γ_B be an arbitrary set of winning paths of player B in game-frame GF_q and consider the game $(V_q, C_q, \emptyset, \Gamma_B)$. From the point of view of the theory of programming it is of great significance to consider whether player A has a non-losing strategy in the game $(V_q, C_q, \emptyset, \Gamma_B)$. The meaning of this consideration for the theory depends on what set Γ_B is selected. To illustrate the significance of the existence of a not losing strategy of A we consider the game $(V_q, C_q, \emptyset, \Gamma_B)$ with different Γ_B.

A. Let Γ_B be the set of those definable paths of the game-frame GF_q in which there exists a situation (node) v such that $l(v) \notin \{i_m \mid m \leqq n\}$, i.e. Γ_B consists of the terminating definable paths of V_q.

In this case "to have a not losing strategy of the player A" means that A is able to ensure that the program execution never "dies", i.e. it runs infinitely long (more precisely ζ-long). □

B. Let $\psi \in F_\vartheta^{X_P}$ be an arbitrary formula that is called the output condition and let Γ_B be the set of those definable paths of V_q each of that contains a node v such that

(i) $l(v) \notin \{i_m \mid m \leqq n\}$ and
(ii) $\mathfrak{A} \not\models \psi[s(v)]$.

So Γ_B consists of those executions of the program p that terminate without satisfying the output condition ψ.

In this case the non-losing strategy of A means the partial correctness of the program p with respect to the output condition ψ (while the input condition is any tautology e.g. $x = x$). □

C. Let $\varphi, \psi \in F_S^{X_p}$ be arbitrary formulas that are called input and output conditions respectively and let Γ_B be the set of those definable paths of V_q for each of which:

(i) $\mathfrak{A} \models \varphi[s(0)]$,

(ii) there exists a node (in the path) such that

$$l(v) \notin \{i_m | m \leqq n\} \quad \text{and} \quad \mathfrak{A} \not\models \psi[s(v)].$$

In this case the non-losing strategy of A means the partial correctness of the program p w.r.t. the input condition φ and output condition ψ. Note that we can suppose that if the program executed by input q which does not satisfy the condition φ then $\Gamma_B = \emptyset$. □

D. Let $\varphi, \psi \in F_S^{X_p}$ be arbitrary formulas and let Γ_B be the set of definable paths of V_q such that each satisfies both

(i) $\mathfrak{A} \models \varphi[s(0)]$ and

(ii) either it does not terminate or it has a node v such that $l(v) \notin \{i_m | m \leqq n\}$ and $\mathfrak{A} \not\models \psi[s(v)]$.

Now the non-losing strategy of A means the total correctness of the program p w.r.t. the input condition φ and output condition ψ. □

Before proceeding to the next cases we introduce further notations. The value of the input is subject to change while executing a program, i.e. during the corresponding computing process. Thus in order to describe program properties during execution we need both the input values and the actual values of program variables. For any fixed program p we distinguish a set $X_p \subset Y$ of variable symbols that duplicates the variable symbols Y_p and refers to the input values of the latter.

If $Y_p = \{y_1, \ldots, y_k\}$ then $X_p = \{x_1, \ldots, x_k\}$ and for any $i \in [0, n]$, x_i duplicates the corresponding y_i.

Moreover let t be a distinguished variable symbol of Y which will be used to describe time conditions. A formula $\varphi \in F_{S^*}^Y$ is said to be an input condition for a fixed program p iff $\operatorname{Var} \varphi \subset X_p$. Moreover, a formula $\psi \subset F_{S^*}^Y$ is called output condition for the program p iff $\operatorname{Var} \psi \subset X_p \cup Y_p \cup \{t\}$.

Now let us consider the further cases.

E. Let φ and ψ be input and output conditions for a fixed program p respectively and let Γ_B be the set of those definable paths of V_q which satisfy $\mathfrak{A} \models \varphi[q]$ and have a node v such that $l(v) \notin \{i_m | m \leqq n\}$ and $\mathfrak{A} \not\models \psi[q, s(v), \mathit{length}(v)]$.

In this case the existence of a non-losing strategy of player A is of the following meaning. An execution of program p carried out in accordance whith the strategy is such that it starts with input q that satisfies condition φ and when it stops it satisfies condition ψ. □

F. Let φ and ψ be input and output conditions for program p respectively and let Γ_B be the set of such definable paths of V_q that each of them satisfies $\mathfrak{A} \models \varphi[q]$ and it is either infinite or there exists a node v such that $l(v) \in \{i_m | m \leqq n\}$ and $\mathfrak{A} \not\models \not\models \psi[q, s(v), \mathit{length}(v)]$.

In this case the existence of a non-losing strategy of player A means the following. An execution of program p done in accordance with the strategy starts with input q satisfying condition φ and it terminates by satisfying condition ψ. □

We could go on listing cases infinitely, but we hope that the aboves represent the basic idea well.

However each of the above cases represents its own situation in the theory of programming, having its application in the mentioned way. From theoretical point of view it would be very useful to have a unique descriptive language for any version of the set Γ_B, i.e. this language has to be suitable to define sets of paths in trees. But for the time being we have not such a universal descriptive language. Thus for each Γ_B we have to introduce a new descriptive language remaining within the same definable game-frame (V_q, C_q).

Corresponding to aboves a theory of nondeterministic programming though having one programming language has to possess several descriptive languages each of which fits in to certain conditions corresponding to some pragmatical aims.

To illustrate this theory we give the appropriate descriptive language for the cases E and F. This language will be common for both cases. We have chosen these cases because in literature time consideration is hardly investigated.

First we introduce the syntax of the descriptive language.

Definition 6.1. $\mathbf{F}_\Im \overset{\mathrm{d}}{=} \{(\varphi, p, \psi) \mid p \in NP_\Im,\ \varphi$ and ψ are input and output conditions for p respectively$\} \cup \{[\varphi, p, \psi] \mid p \in NP_\Im,\ \varphi$ and ψ are input and output conditions for p respectively$\}$. □

When we are interested only in the triple of φ, p, ψ without specifying the type of brackets that include them we write $|\varphi, p, \psi|$.

Now we are going to define the semantics of the descriptive language.

Let $\chi = |\varphi, p, \psi| \in \mathbf{F}_\Im$ and let $\mathfrak{A} \in Md(Ax^*)$. According to Lemma 5.4 for any q: Var $p \to A$ there is a game-frame $G_p = (V_q, C_q)$ and a function $f_q = (l, s)$: $V_q \to A$ and they are definable in Ax^*. Starting out from the game-frame G_q and taking the formula χ into account the following game can be constructed:

(i) $\Gamma^\chi_A = \emptyset$,

(ii) if $\mathfrak{A} \not\models \varphi[q]$ then let $\Gamma^\chi_B = \emptyset$. In opposite case there are two possibilities:

a) First if $\chi = (\varphi, p, \psi)$ then $\Gamma^\chi_B \overset{\mathrm{d}}{=} \{\gamma$ is a definable path in $V_q \mid$ there is a $v \in \gamma$ such that $l(v) \notin \{i_m \mid m \leqq n\}$ and $\mathfrak{A} \not\models \psi[q, s(v), \mathit{length}\,(v)]\}$.

b) Secondly if $\chi = [\varphi, p, \psi]$ then let $\Gamma^\chi_B \overset{\mathrm{d}}{=} \{\gamma$ is a definable path in $V_q \mid \gamma$ is either infinite or there is a $v \in \gamma$ such that $l(v) \notin \{i_m \mid m \leqq n\}$ and $\mathfrak{A} \not\models \psi[q, s(v), \mathit{length}\,(v)]\}$.

Thus we have got a game $G^\chi_q = (V_q, C_q, \emptyset, \Gamma^\chi_B)$ which is called *the associated q-game generated by* χ.

By using the technique used in the proof of Theorem 4.12 it is not difficult to verify that the set Γ^χ_B is parametrically definable in Ax^* in both of the above cases a) and b). Moreover, the value of q are used only as parameters in the defining formulas. Thus we have the following

Lemma 6.2. Let $\mathfrak{A} \in Md(Ax^*)$ be arbitrary, $\chi \in \mathbf{F}_\Im$ and q: $X_p \to A$. Then the associated q-game $(V_q, C_q, \emptyset, \Gamma^\chi_B)$ is parametrically definable in Ax^*. □

Now we turn to the definition of the semantics of the descriptive language to be defined.

Definition 6.3. Let $\mathfrak{A} \in Md(Ax^*)$ and $\chi \in \mathbf{F}_\vartheta$ be arbitrary. $\mathfrak{A} \models\!\equiv \chi$ iff for any q: Var $p \to A$ player A has a non-losing definable strategy in the associated q-game G_q^χ. □

Note that if $\chi = (\varphi, p, \psi)$ ($\chi = [\varphi, p, \psi]$) then $\mathfrak{A} \models\!\equiv \chi$ means the partial (respectively total) correctness of the program p in the model $\mathfrak{A}$ w.r.t. the input condition φ and output condition ψ.

Definition 6.4. For an arbitrary ϑ-type system Ax^* the descriptive language expressing partial and total correctness and describing the corresponding time condition is

$$\mathbf{D}_\vartheta^{Ax^*} \overset{\mathrm{d}}{=} (\mathbf{F}_\vartheta, Md(Ax^*), \models\!\equiv).$$

Remark 6.5. Similarly to the fact that the associated q-game is parametrically definable, if $\mathfrak{A} \models\!\equiv \chi$ then the non-losing strategy and the corresponding run are also parametrically definable in $\mathfrak{A}$.

What is the connection between the validity relation introduced above and the validity relation of Definition 3.5. The answer to this and the expressibility of some notions introduced above by the first order language is given by the following

Theorem 6.6. For any $\chi \in \mathbf{F}_\vartheta$ there is a formula $\chi^* \in F_{\vartheta *}^X$ such that for any $\mathfrak{A} \in Md(Ax^*)$, $\mathfrak{A} \models\!\equiv \chi$ iff $\mathfrak{A} \models \chi^*$.

The function rendering χ^* to χ is recursive.

Moreover, there is a formula $\sigma_\chi(\vec{a}, v, w) \in F_{\vartheta *}^X$ such that if $\mathfrak{A} \models\!\equiv \chi$ then for any q: Var $p \to A$, $\sigma_\chi[q]$ defines a non-losing strategy of player A in the associated q-game G_q^χ.

Proof. The existence of the formula χ^* for a given $\chi \in \mathbf{F}_\vartheta$ is an easy consequence of Theorem 4.9. The recursiveness of the function that renders the corresponding χ^* to each of them can be established by using Gödel-numbering. We omit the details. □

Let $E_0^{Ax^*}$ denote the set of all formulas that are true in the models of Ax^*, i.e. $E_0^{Ax} \overset{\mathrm{d}}{=} \{\chi \mid Ax^* \models\!\equiv \chi\}$.

From the point of view of the usability of the descriptive language $\mathbf{D}_\vartheta^{Ax^*}$ for the theory of programming the handlebility of $E_0^{Ax^*}$ is very important. This is established by the following

Theorem 6.7. (Completeness.) If the system Ax^* is recursively enumerable then so is set of formulas $E_0^{Ax^*}$, i.e. the descriptive language $\mathbf{D}_\vartheta^{Ax^*}$ is complete.

Proof. Immediate from 6.6. □

Remark 6.8. If $\mathfrak{A} \models\!\equiv \chi$ would be defined so that for any input data q in the game G_q^χ player A had a non-losing strategy (without the assumption of definability) then the language $\mathbf{D}_\vartheta^{Ax^*}$ would lose the completeness, even more, it would be neither ω-complete nor complete relative to arithmetic in contrast with the case of deterministic programming (see BANACHOWSKI et al (1977) and COOK (1978), PRATT (1978) respectively).

7. A complete calculus

Being aware of the completeness of the language $\mathbf{D}_{\vartheta}^{Ax^*}$ we are going to introduce such a complete calculus which is close to the programmer's intuition. This calculus has to be convenient to prove about a given program p either its partial or total correctness w.r.t. given input and output conditions. To provide such a calculus is significant from the point of view of both theory and practice.

Theorem 6.6 shows that the syntax $\mathbf{F}_{\vartheta}$ can be coded in the syntax F_{ϑ}^{Y} however the proofs in the latter can not be interpreted directly by programming situations.

Below we introduce a calculus in the spirit of Floyd and Hoare.

First we introduce the following

Notation 7.1. Let $Lab\colon NP_{\vartheta} \to P(\mathbf{N})$ be the function rendering to each nondeterministic program $p=(i_0\colon u_0, \ldots, i_n\colon u_n) \in NP_{\vartheta}$ the set of labels occuring in it as follows.

$Lab\,(p)=\{i_m \mid m \leqq n+1\} \cup \{k \mid$ there is an m such that $u_m = \textit{if } \varkappa \textit{ then } \square \ldots k \ldots \square \in \{\text{any, every}\}\}$.

Definition 7.2. Let us fix a ϑ-type system Ax^* and let $\chi = |\varphi, p, \psi|$ be an arbitrary formula of $\mathbf{F}_{\vartheta}$. Let $\Phi\colon Lab\,(p) \to F_{\vartheta^*}$ be a function which to each label of the program p renders a formula $\Phi(i)$ with variable symbols $\vec{x}, \vec{y}, t \in Y$. Remember that $\vec{x}$ serves to duplicate the program variable $\vec{y}$ and t refers to time. Further on we write Φ_i instead of $\Phi(i)$. The function Φ is said to be the description of χ w.r.t. Ax^* iff it satisfies the following conditions:

(i) $Ax^* \models \varphi \to \Phi_{i_0}[0/t, \vec{x}/\vec{y}]$ (input rule)

(ii) if $i_m\colon y \leftarrow \tau$ then

$Ax^* \models \Phi_{i_m} \to \Phi_{i_m+1}[\tau/y, t+1/t]$ (assignment rule)

(iii) if $i_m\colon \textit{any } y \leqq \tau$ then

$Ax^* \models \Phi_{i_m} \to \exists z\,(z \leqq \tau \wedge \Phi_{i_m+1}[z/y, t+1/t])$ (assignment rule of A's choice)

(iv) if $i_m\colon \textit{every } y \leqq \tau$ then

$Ax^* \models \Phi_{i_m} \to \forall z\,(z \leqq \tau \wedge \Phi_{i_m+1}[z/y, t+1/t])$ (assignment rule of B's choice)

(v) if $i_m\colon \textit{if } \varkappa \textit{ then any } k_1, \ldots, k_r$ then

$$Ax^* \models \Phi_{i_m} \wedge \neg\varkappa \to \Phi_{i_m+1}[t+1/t], \quad Ax^* \models \Phi_{i_m} \wedge \varkappa \to \bigvee_{j=1}^{r} \Phi_{k_j}[t+1/t]$$

(rule of conditional jump of A's choice)

(vi) if $i_m\colon \textit{if } \varkappa \textit{ then every } k_1, \ldots, k_r$ then

$$Ax^* \models \Phi_{i_m} \wedge \neg\varkappa \to \Phi_{i_m+1}[t+1/t], \quad Ax^* \models \Phi_{i_m} \wedge \varkappa \to \bigvee_{j=1}^{r} \Phi_{k_j}[t+1/t]$$

(rule of conditional jump of B's choice)

(vii) $Ax^* \models \Phi_z \to \psi$ (output rule for $z \notin \{i_m \mid m \leqq n\}$)

Moreover if $\chi=[\varphi, p, \psi]$ then
(viii) for any $z\in\{i_m \mid m\leqq n\}$
$Ax^* \models \exists v\, \forall \bar{y}[\Phi_z \to v \geqq t]$ (rule of termination).
We denote by $Ax^* \vdash \chi$ the fact that the formula χ has a description w.r.t. Ax^*. □

Now let us see some remarks on the above definition:

Each formula Φ_i $(0\leqq i\leqq n)$ shows the conditions the data should satisfy before executing the command labelled by i and the corresponding time conditions. We note that ζ can be used in any formula Φ_z.

The rule (i) shows that the execution of the program starts at the moment 0.

In the rules (ii)—(vi) the substitution $[t+1/t]$ denotes unambiguously that the execution of each command happens during one unit of time however complex e.g. the term τ is. This has already been supposed in the definition of the q-game associated to the program (see 5.3).

This assumption can be generalized without any trouble in the following way. We render to any command i_m: u_m an execution time t_m which can also depend on the data $\bar{y}$. Thus the game associated to the program continues a unique path during t_m time-units. Now this corresponds to the fact that the game stays in one and the same state. Using this generalization in the description of χ in every rule (ii)—(vi) instead of the substitution $[t+1/t]$ we write $[t+t_m/t]$.

The rule (vii) says that if the execution of the program p stops then it stops forever i.e. its time stops — the process "dies".

The rule (viii) is needed only to prove the total correctness of p.

In the descriptive formulas the usage of separate variable symbols x is allowed in order to refer to the input values. These are needed only in the formulas used in the description of total correctness. While proving the completeness we shall get a formula independent from x for the case $\chi=(\varphi, p, \psi)$, by using the definition $\Phi_z^* \overset{d}{=} \exists x \Phi_z$ for each descriptive formula Φ_z.

It is interesting that in the case of nondeterministic programming total correctness is not equal to partial correctness plus termination in contrast with the deterministic sequential case. Thus in our case the total correctness cannot be established by proving the partial correctness and the termination separately.

Now we show that the calculus introduced above is complete.

Theorem 7.3. (Completeness.) For any ϑ-type system Ax^* and $\chi\in\mathbf{F}_\vartheta$, $Ax^*\models\chi$ iff $Ax^*\vdash\chi$.

Proof. First we show that if $Ax^*\vdash\chi$ then $Ax^*\models\chi$.

Let us fix a $\chi=|\varphi, p, \psi|$ and let Φ: $Lab\, p \to F_{\vartheta *}^X$ be a description of χ w.r.t. Ax^*. Let $\mathfrak{A}\in Md(Ax^*)$ and q: $\mathrm{Var}\, p \to A$ be arbitrary. We must prove that in the q-game $G_q^\chi=(V_q, C_q, \emptyset, \Gamma_B^\chi)$ player A has a non-losing strategy.

Let (l, s): $V_q \to \mathbf{N}\times{}^{\mathrm{Var}\, p}A$ be the trace of p in $\mathfrak{A}$ starting with q. First we define a strategy for player A.

Let $v\in C_q$ be arbitrary. Since $v\in C_q$ for an $m\in[1, n]$ we have $l(v)=i_m$ and $u_m=$ $=any\ y\leqq\tau$ or $u_m=if\ \varkappa\ then\ any\ k_1, \ldots, k_r$. If $\mathfrak{A}\models\Phi_{l(v)}[q, s(v), length\ v]$ then let $str\,(v)$ be the minimal element of $S_{V_q}(v)$. (It might be arbitrary but we have to be

sure that *str* is parametrically definable.) So let us suppose that $\mathfrak{A} \models \Phi_{l(v)}[q, s(v), \mathit{length}(v)]$. First let $u_m = \mathit{any}\ y \leqq \tau$. By our assumption with $i_m = l(v)$

$$\mathfrak{A} \models \Phi_{i_m} \to \exists z (z \leqq \tau \wedge \Phi_{i_{m+1}}[\tau/y, t+1/t],$$

$\mathfrak{A} \models \Phi_{i_m}[q, s(v), \mathit{length}\, v]$ and thus

$$\mathfrak{A} \models \exists z (z \leqq \tau \wedge \Phi_{i_{m+1}}[\tau/y, t+1/t][q, s(v), \mathit{length}(v)].$$

Now let z^* be the minimal $\underset{\sim}{z} \in A$ such that

$$\mathfrak{A} \models \underset{\sim}{z} \leqq \tau \wedge \Phi_{i_{m+1}}[\tau/y, t+1/t][q, s(v), \mathit{length}(v), \underset{\sim}{z}].$$

It is clear that $\mathit{pair}(v, z^*) \in S_{V_q}(v)$ and so let

$$\mathit{str}(v) \overset{\mathrm{d}}{=} \mathit{pair}(v, z^*).$$

Obviously

$$\mathfrak{A} \models \Phi_{i_{m+1}}[q, s(\mathit{str}(v)), \mathit{length}(\mathit{str}(v))] \tag{1}$$

In the end let $u_m = \mathit{if}\ \varkappa\ \mathit{then\ any}\ k_1, \ldots, k_r$. Since $v \in C_q$ so $\mathfrak{A} \models \varkappa[s(v)]$. Using $\mathfrak{A} \models \Phi_{i_m} \wedge \varkappa \to \bigvee_{j=1}^{r} \Phi_{k_j}[t+1/t]$ and $\mathfrak{A} \models \Phi_{i_m} \wedge \varkappa[q, s(v), \mathit{length}(v)]$ we have

$$\mathfrak{A} \models \bigvee_{j=1}^{r} \Phi_{k_j}[t+1/t][q, s(v), \mathit{length}(v)].$$

Let k^* be the minimal $k \in \{k_1, \ldots, k_r\}$ for which

$$\mathfrak{A} \models \Phi_k[t+1/t][q, s(v), \mathit{length}(v)] \tag{2}$$

and take $\mathit{str}(v) \overset{\mathrm{d}}{=} \mathit{pair}(v, k^*)$.

Thus the strategy *str* is defined. Since G_q is definable and the method used in the aboves can be defined in $F_{3^*}^X$ so *str* is also parametrically definable. The fact, that *str* is a non-losing strategy follows from the following

Lemma 7.4. Let π be a parametrically definable path in the run R_{str} of the strategy *str*. For every $v \in \pi$

$$\mathfrak{A} \models \Phi_{l(v)}[q, s(v), \mathit{length}(v)].$$

In fact if $l(v) \notin \{i_m \mid m \leqq n\}$ then

$$\mathfrak{A} \models \psi[q, s(v), \mathit{length}(v)].$$

Moreover, if $\chi = [\varphi, p, \psi]$ then π terminates.

Proof. The proof goes by induction on the *length* (v). If $v = \Lambda$ and $\mathit{length}(v) = 0$ then by using $\mathfrak{A} \models \varphi[q]$ and (i) of Definition 7.2, $\mathfrak{A} \models \Phi_{l(\Lambda)}[q, s(\Lambda), 0]$. The inductive step can be done by using (1), (2) and (ii)—(vi) of Definition 7.2. If $l(v) \notin \{i_m \mid m \leqq n\}$ then from (vii) of Definition 7.2 we get $\mathfrak{A} \models \psi[q, s(v), \mathit{length}(v)]$.

If $\chi = [\varphi, p, \psi]$ then let us apply (viii) of Definition 7.2 and we get a $\underset{\sim}{t} \in A$ such that if $\mathit{length}(v) > \underset{\sim}{t}$ then for any $z \in \{i_m \mid m \leqq n\}$, $\mathfrak{A} \models \Phi_z[q, s(v), \mathit{length}(v)]$. Hence using $\mathfrak{A} \models \Phi_{l(v)}[q, s(v), \mathit{length}(v)]$ we have $l(v) \notin \{i_m \mid m \leqq n\}$.

The proof of the lemma is completed. □

Now the proof of the theorem is continued by showing that if $Ax^* \models \chi$ then $Ax^* \vdash \chi$.

Let $\chi = |\varphi, p, \psi|$ be such that $Ax^* \models \chi$. Let $\Omega_V(\vec{a}, v)$ and $\Omega_C(\vec{a}, w)$ be the formulas which parametrically define the game-frame $GF = (V, C)$ in Ax^* generated by χ. By the second part of Theorem 6.6 there is a formula $\sigma_\chi(\vec{a}, v, w)$ with the same parameters as Ω_V, Ω_C which in Ax^* parametrically defines a non-losing strategy for player A in the game G_q^χ. Hence there is a formula $\varrho_\chi(\vec{a}, r) \in F_{\mathfrak{I}^*}^X$ which defines in Ax^* the run of the strategy defined by σ_χ. Now we give a description of χ in Ax^*:

$$\Phi_z(\vec{x}, \vec{y}, t) \overset{d}{=} \varphi(\vec{x}) \wedge \exists v \{\varrho_\chi(\vec{x}, v) \wedge l(v) = z \wedge \vec{y} = s(v) \wedge length(v) = t\}.$$

Since s is parametrically definable in Ax^* by using only the parameter $\vec{x}$ so $\Phi_z \in F_{\mathfrak{I}^*}^X$. It is clear that $\operatorname{Var} \Phi_z \subset \{x_1, \ldots, x_k, y_1, \ldots, y_k, t\}$.

The following lemma is enough to complete the proof.

Lemma 7.5. The above function Φ is a description of χ w.r.t. Ax^*.

Proof. Let $\mathfrak{A} \in Md(Ax^*)$ and q: $\operatorname{Var} p \to A$ be arbitrary. We have to prove that for any $\vec{y}, \underset{\sim}{t} \in A$ the rules in Definition 7.2 hold. (i)—(vii) can be proved in the same way so e.g. we prove (iii). Let $\vec{y}, \underset{\sim}{t} \in A$ be such that $\mathfrak{A} \models \Phi_{i_m}[q, \vec{y}, \underset{\sim}{t}]$ and $u_m = every\ y \leqq \tau$. Let $G_q^\chi = (V_q, C_q, \emptyset, \Gamma_B^\chi)$ be the associated q-game generated by χ and let (l, s) be the trace of p in $\mathfrak{A}$. By the definition of Φ_{i_m} there is a $v \in V_q$ such that $\vec{y} = s(v)$ and $t = length(v)$. Moreover $v \in R_{str}$ where str is the non-losing strategy of player A defined by σ and q. Let $\underset{\sim}{z} \leqq \tau_{\mathfrak{A}}[s(v)]$ be arbitrary. Then $w = pair(v, \underset{\sim}{z}) \in R_{str}$, $length(w) = \underset{\sim}{t} + 1$,

$$s(w)(x) = \begin{cases} \underset{\sim}{z} & \text{if } x = y, \\ s(v)(x) & \text{otherwise;} \end{cases}$$

and $l(v) = i_{m+1}$.

This means that

$$\mathfrak{A} \models \Phi_{i_m}[q, s(w), l(w)].$$

Hence

$$\mathfrak{A} \models \Phi_{i_{m+1}}[z/y, t+1/t][q, \vec{y}, \underset{\sim}{t}].$$

Summarizing:

$$\mathfrak{A} \models \Phi_{i_m} \to \forall z (z \leqq \tau \wedge \Phi_{i_{m+1}}[z/y, t+1/t]).$$

If $\chi = [\varphi, p, \psi]$ we must prove that for any $z \notin \{i_m \mid m \leqq n\}$,

$$\mathfrak{A} \models \exists T\ \forall \vec{y}\ \forall t [\Phi_z(\vec{y}, t) \to t \leqq T].$$

By the definition of G_q^χ and R_{str} in R_{str} any definable path terminates. It is clear that R_{str} is finitary and thus König's Lemma can be applied: there is a $\underset{\sim}{T} \in A$ such that for any $v \in R_{str}$, $length(v) \leqq \underset{\sim}{T}$. As in the first part of the proof for any $(\vec{y}, \underset{\sim}{t}) \in A$ with

$$\mathfrak{A} \models \Phi_z[q, \vec{y}, \underset{\sim}{t}]$$

there is a $v \in R_{str}$ such that $s(v) = \vec{y}$ and $length(v) = \underset{\sim}{t}$. Hence $\underset{\sim}{t} \leqq \underset{\sim}{T}$ and thus

$$\mathfrak{A} \models \forall \vec{y}\ \forall t [\Phi_z(\vec{y}, t) \to t \in T][q, \underset{\sim}{T}].$$

The proof of the lemma is completed and so is the proof of the theorem. □

8. Examples

8.1. The least common multiplier

Let us consider the following nondeterministic program that we would like to use for the computation of the least common multiplier of two positive integers:

$p_1 \overset{d}{=}$

0: *any* $y_1 \leqq \max(a, b)$

1: *if* $a|y_2 \wedge b|y_2 \wedge y_2 \neq 0$ *then* 5

2: *if* $y_2 > ab$ *then* 5

3: $y_2 \leftarrow y_2 + y_1$

4: *goto* 1

The input and output conditions are respectively

$$\varphi \overset{d}{=} a > 0 \wedge b > 0 \wedge y_2 = 0$$

and

$$\psi \overset{d}{=} y_2 = [a, b].$$

Let Ax be the set of axioms, which contains PA and the axiomatic definition of $|$, max, $<$, $\leqq$, and $[.,.]$. Let

$$Ax^* \overset{d}{=} Ax \cup \{\forall x \zeta(x)\}.$$

We are going to prove that

$$Ax^* \models (\varphi, p_1, \psi).$$

i.e. in the label 0 the value of y_1 can be chosen so that if the program terminates then $y_2 = [a, b]$. Indeed let us consider the following description of χ w.r.t. Ax^*:

$$\Phi_0 \overset{d}{=} a > 0 \wedge b > 0 \wedge y_2 = 0$$

$$\Phi_1 \overset{d}{=} y_1 > 0 \wedge (y_1|a \vee y_1|b) \wedge \forall z\,(z \leqq y_2 \wedge y_1|z \rightarrow \neg z|a \vee \neg z|b)$$

$$\Phi_2 \overset{d}{=} y_1 > 0 \wedge (y_1|a \vee y_1|b)$$

$$\Phi_3 \overset{d}{=} \Phi_2 \wedge y_2 \leqq ab$$

$$\Phi_4 \overset{d}{=} \Phi_1$$

$$\Phi_5 \overset{d}{=} y_2 = [a, b]$$

It is easy to verify that the function described above is really the description of χ w.r.t. Ax^*.

It is obvious that the greater value is chosen for y_1 from that of satisfying the condition Φ_1 the sooner termination of the program occurs. That is why from

the output condition we claim the expression of early termination. So let

$$\tilde{\psi} \overset{d}{=} y_2 = [a, b] \wedge t \leqq \frac{4([a, b]+1)}{\max(a, b)}.$$

Now let us consider the description $\tilde{\Phi}$ of $\tilde{\chi}=[\varphi, p, \tilde{\psi}]$ w.r.t. Ax^*, where the time conditions are to be expressed in the descriptive formulas $\tilde{\Phi}_i$ $(0 \leqq i \leqq 5)$. The description $\tilde{\Phi}$ is a light modification of Φ as follows:

$$\tilde{\Phi}_0 \overset{d}{=} a > 0 \wedge b > 0 \wedge y_2 = 0$$

$$\tilde{\Phi}_1 \overset{d}{=} y_1 = \max(a, b) \wedge y_2 = \frac{t-1}{4} y_1 \wedge y_2 \leqq ab \wedge \forall z (z \leqq y_2 \wedge y_1 | z \rightarrow \neg z | a \vee \neg z | b)$$

$$\tilde{\Phi}_2 \overset{d}{=} y_1 = \max(a, b) \wedge y_2 = \frac{t-2}{4} y_1 \wedge y_2 \leqq ab \wedge \forall z (z \leqq y_2 \wedge y_1 | z \rightarrow \neg z | a \vee \neg z | b)$$

$$\tilde{\Phi}_3 \overset{d}{=} \tilde{\Phi}_2[t-3/t]$$

$$\tilde{\Phi}_4 \overset{d}{=} \tilde{\Phi}_1[t+1/t]$$

$$\tilde{\Phi}_5 \overset{d}{=} y_2 = [a, b] \wedge t \leqq \frac{4([a, b]+1)}{\max(a, b)}.$$

From the descriptions Φ and $\tilde{\Phi}$ it immediately follows that the command 2: *if* $y_2 > ab$ *then* 5 is really unnecessary. It is good exercise to prove that if in the program p_1 we write *every* instead of each occurence of *any* we get a program p_1' such that

$$Ax^* \models [\varphi, p_1', t \leqq 4ab+4].$$

8.2 Pattern matching

Now we show how the nondeterministic programming language NP_ϑ can be used for handling the problem of pattern matching. First let us specify what pattern-matching is.

Let $\leqq$ be a definable partial ordering in a fixed system Ax^*. Let $\mathfrak{A} \in Md(Ax^*)$. If for some $a, b \in A$, $a \leqq b$ then we say that b is defined at least that much as a. Let us give a vector $\langle X(j) \rangle_{j \leqq n}$ of $n+1$ elements. We would like to match to this vector a pattern from the matrix $\langle A(i, j) \rangle_{i \leqq m, j \leqq n}$.

More precisely, we would like to find such a row in the matrix each element of which is defined at least that much as the corresponding element of the vector $\langle X(j) \rangle_{j \leqq n}$.

Since vectors and matrices are not allowed in our language NP_ϑ explicitly we have to suppose that the type ϑ contains the function symbols *vcomp* and *mcomp*, for which $\vartheta(vcomp)=2$ and $\vartheta(mcomp)=3$. The function symbol *vcomp* provides the i-th component of the vector having the code $\bar{X}$. Analoguously *mcomp* provides the (i, j)-th component of the matrix having the code $\bar{A}$.

Let us consider the following program the input data of which are A, x, n, m.

$p_2 \overset{d}{=}$

0: *any* $i \leqq m$
1: *every* $j \leqq n$
2: *if* $vcomp(x, j) \leqq mcomp(A, i, j)$ *then* 5
3: $u=0$
4: *goto* 6
5: $u=1$

Let $u=1$ be the output condition of the program. It is a useful exercise to prove the fact that the execution of p_2 w.r.t. the input values $\bar{A}, x, n, m$ is correct iff such a row s can be chosen for the label 0 that can be matched to $\bar{X}$.

8.3 A NIM-game

According to the definition the execution of programs is represented by an associated game. Of course this can be reversed as follows: the properties of a game can be represented by an appropriate nondeterministic program.

Let us consider the following version of the game NIM. There are two players A and B and there is a single pile of n chips. The two players take turns alternately and at each move a player must pick up at least one and at most k chips from the pile. This is the game-frame. We add to this the following: the player who picks up the last chip wins. Thus we have the game G.

Let player A move first. By using the calculus of Definition 7.2 and Theorem 7.3 we show that player A has a winning strategy if n is divisible by $k+1$ i.e. if $k+1 | n$.

Let us take the same system Ax^* as in 8.1 and consider the following program.

$p_3 \overset{d}{=}$

0: *any* $y \leqq k$
1: *if* $y=0$ *then* 9
2: *if* $y \geqq n$ *then* 11
3: $n \leftarrow n-y$
4: *every* $y \leqq k$
5: *if* $y=0$ *then* 11
6: *if* $y \geqq n$ *then* 9
7: $n \leftarrow n-y$
8: *goto* 0
9: $u=0$
10: *goto* 12
11: $u=1$

Note that the commands of labels 0 and 4 allow to choose 0 chips in contrast with the rules of the game G. This is connected with our programming language NP_3, where for the sake of simplicity we introduced the assignment commands with choices in the form of i: $\square y \leqq t$, $\square \in \{any, every\}$. However we could have given them in the following form i: $\square\ \tau_1 \leqq y \leqq \tau_2$ or i: $\square\ y \in \varphi$ (where $\tau_1, \tau_2 \in T_3^Y$ and $\varphi \in F_3^Y$ and it has at least one free variable symbol). Thus to keep up with the game rules we need the commands labelled by 1 and 5. These commands assure that if one of the players on his turn chooses 0 chips that this move will be considered as a cheat and he loses the game immediately.

It is obvious that player A has winning strategy in the game G iff the program p_3 is correct w.r.t. the output condition $\psi \overset{d}{=} u = 1$. Since in the game G drow is not possible so a non-losing strategy of player A is, at the same time, his winning strategy. Thus it is enough to show that $Ax^* \vdash \chi$ where

$$\chi = [\neg k+1 | n, p_3, u = 1],$$

i.e. it is enough to give a description of the formula χ w.r.t. Ax^*. It is easy to verify that the formulas below define a description of χ w.r.t. Ax^*:

$$\Phi_0 \overset{d}{=} \neg k+1 | n$$

$$\Phi_1 \overset{d}{=} 1 \leqq y \leqq k \wedge y \equiv n \bmod (k+1)$$

$$\Phi_2 \overset{d}{=} 1 \leqq y \leqq k \wedge y \equiv n \bmod (k+1)$$

$$\Phi_3 \overset{d}{=} y < n \wedge y \equiv n \bmod (k+1)$$

$$\Phi_4 \overset{d}{=} k+1 | n \wedge n > 0$$

$$\Phi_5 \overset{d}{=} y \leqq k \wedge n > 0 \wedge k+1 | n$$

$$\Phi_6 \overset{d}{=} 1 \leqq y \leqq k \wedge k+1 | n \wedge n > 0$$

$$\Phi_7 \overset{d}{=} 1 \leqq y \leqq k \wedge k+1 | n \wedge n > 0$$

$$\Phi_8 \overset{d}{=} \neg k+1 | n$$

$$\Phi_9 \overset{d}{=} false$$

$$\Phi_{10} \overset{d}{=} false$$

$$\Phi_{11} \overset{d}{=} true$$

$$\Phi_{12} \overset{d}{=} u = 1.$$

Abstract (to Part 2)

Games are adequate to nondeterministic programming shown in Part 1 and the theory of nondeterministic programming is intended to be developed within the frame of classical first order logic. So in Section 4 the representation of games is given by considering definable games within this frame. The nondeterministic programming language is introduced in Section 5. In Section 6 a descriptive language is introduced which, beside the classical data consideration, handles time conditions as well. It is shown (in Theorem 6.7) that this language is complete. Section 7 contains a calculus in the spirit of Floyd and Hoare, the usage of which is illustrated in the last section by examples.

RESEARCH INSTITUTE FOR
APPLIED COMPUTER SCIENCE
CSALOGÁNY U. 30—32.
BUDAPEST, HUNGARY
H—1536

References

[1] BANACHOWSKI, L., A. KRECZMAR, G. MIRKOWSKA, H. RASIOWA and A. SALWICKI, An introduction to algorithmic logic, *Mathematical Foundations of Computer Science*, Banach Center Publications, Warszawa, v. 2, 1977, pp. 7—99.

[2] COOK, S. A., Soundness and completeness of an axiom system for program verification, *SIAM J. Comput.* v. 7, 1978, pp. 73—92.

[3] GERGELY, T. and L. ÚRY, Nondeterministic programming within the frame of first order classical logic, Part 1, *Acta Cybernet.*, v. 4, 1980, pp. 333—354.

[4] HAREL, D., Arithmetical completeness in logics of programs, *Automata, Languages and Programming*, Eds, G. Ausiello and C. Böhm, Springer-Verlag, Heidelberg, 1978.

[5] MENDELSON, E., *Introduction to mathematical logic*, Van Nostrand, N. Y. 1964.

(Received Sept. 7, 1979)