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

By T. Gergely and L. Ury

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

\[
\text{left} (\text{pair} (x, y)) = x \quad \text{and} \quad \text{right} (\text{pair} (x, y)) = y.
\]

Remember that in any model \( M \in Md (A \times^*) \) induction can be done by inner time \( T_M \) i.e. by the set \( \{ t \in A \mid M \models \xi^* [t] \} \) (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 \( M \in Md (A \times^*) \) and let \( D \) be an initial segment of inner time \( T_M \). A function \( s : D \rightarrow A \) is a finite (in \( M \)) sequence. If \( D = [0, n - 1] \) then \( s \) is called an \( n \)-long sequence. A function \( s : T_M \rightarrow A \) is said to be a sequence or a \( \zeta \)-long sequence.
Let $K \subseteq A \setminus \{0\}$ be a definable set and let $^*K$ denote the set of all parametrically definable finite (in $\mathcal{A}$) sequences containing only elements of $K$. Let $A$ denote the empty sequence.

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

**Proof.** Let $\varphi$ be the formula which defines $K$ in $\mathcal{A}$, i.e. $\text{Var } \varphi = \{x\}$ and $\mathcal{A} \models \varphi[a] \iff a \in K$.

Let us consider the following formula:

$$\varphi^*(s) \equiv \exists t (\text{left}(s) \equiv t \land \xi(t) \land \forall i (i \equiv \text{left}(s) \rightarrow \varphi[\Gamma(\text{right}(s), \text{left}(s), i)/x])),$$

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

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

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

Let us take the following sequence:

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

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

Let $f: [0, n] \rightarrow K$ be a parametrically definable function in $\mathcal{A}$. 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 $s \equiv \text{pair}(c, b)$. If $f$ is finite in $\mathcal{A}$ then by using the fact that $Ax^*$ is a system and the definition of $\Gamma$ we have $\mathcal{A} \models \zeta^*[s]$ i.e. there is a $t \in A$ such that $s \equiv t$ and $\mathcal{A} \models \zeta^*[t]$. Thus $\mathcal{A} \models \varphi^*[s]$ and $s$ codes the given function $f$. \(\square\)

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

- pair: $^*K \times K \rightarrow ^*K$,
- left: $^*K \rightarrow ^*K$,
- right: $^*K \rightarrow A$. \(\square\)

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 $A$ 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.**

$$\text{length}(a) = \begin{cases} 0 & \text{if } a = A \\ \text{length}(\text{left}(a)) + 1 & \text{otherwise} \end{cases}$$
Definition 4.5. A set \( V \subseteq \mathbb{A} \setminus \{0\} \) is called a tree iff
1. \( A \subseteq V \),
2. \( \forall v \in V \Rightarrow \text{left}(v) \in V \).

For any \( V \) and \( \bar{v} \in V \) take \( S_{V}(v) = \{ v \in V \mid \text{left}(v) = \bar{v} \} \).

A tree \( V \) is called a path iff \( \text{left} : V \setminus \{A\} \rightarrow V \) is an injection, i.e. any \( v \in V \) has at most one successor.

A tree \( V \) is definable in \( \mathcal{A} \) iff it is parametrically definable in \( \mathcal{A} \) as a unary relation on \( A \).

Note that because \( 0 \) codes \( A \) it is sufficient to use \( \mathbb{A} \setminus \{0\} \) 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 \( \mathcal{A} \) is a function of the form \( f : V \rightarrow ^{n}A \) (for some natural number \( n \)) where \( V \) is a tree in \( \mathcal{A} \). If \( f \) is parametrically definable then it said to be a definable trace in \( \mathcal{A} \).

Definition 4.7. Let \( V \) be a trace in \( \mathcal{A} \). A game-frame in \( \mathcal{A} \) is a pair \( GF=(V, C) \) where \( C \subseteq V \) 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 \( R \subseteq V \) such that
1. if \( c \in C \cap R \) then \( c \) has a unique successor in \( R \),
2. if \( r \in R \setminus C \) then \( S_{V}(r) = S_{R}(r) \).

A strategy in a game-frame \( G \) is a function \( str : C \rightarrow \text{rg} \text{str} \) in such a way that for any \( c \in C, str(c) \in S_{V}(c) \).

A game-frame is definable in \( \mathcal{A} \) iff \( V \) and \( C \) are parametrically definable. A run \( R \) of \( GF \) is definable in \( \mathcal{A} \) if both \( GF \) and \( R \) are parametrically definable in \( \mathcal{A} \).

A strategy \( str \) in \( GF \) is definable in \( \mathcal{A} \) 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 \( \mathcal{A} \) is finitary iff for any \( v \in V \) there is a \( d \in \mathbb{A} \) such that
1. \( \mathcal{A} = \mathbb{A} \setminus \{d\} \),
2. if pair \( (v, e) \in S_{V}(v) \) then \( e \equiv d \), i.e. any node of \( V \) has only finitely many successors (in \( \mathcal{A} \)).

Definition 4.10. A game in \( \mathcal{A} \) is a quadruple \( \mathcal{G}=(V, C, \Gamma_{A}, \Gamma_{B}) \) where \( (V, C) \) is a game-frame and \( \Gamma_{A}, \Gamma_{B} \) are disjoint sets of paths in \( V \).

A game \( (V, C, \Gamma_{A}, \Gamma_{B}) \) is definable in \( \mathcal{A} \) iff
1. \( (V, C) \) is a definable game-frame,
2. \( \Gamma_{A} \) and \( \Gamma_{B} \) are definable sets,
3. each path of \( \Gamma_{A} \) and \( \Gamma_{B} \) 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 \( \Gamma_{A} \) (not to \( \Gamma_{B} \)).
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 \( \Gamma_B \) and \( R \).

Thus a strategy 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 \( \Gamma_A \) (not belonging to \( \Gamma_B \)). 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 \( \exists \)-type system \( Ax^* \). There is a recursive map \( F^*_\exists \rightarrow N \) (\( g \rightarrow \Gamma \varphi \) where \( \Gamma \varphi \) denotes the Gödel number of the formula \( \varphi \)) and a formula \( \text{Valid}(g, a, x) \in F^*_\exists \) such that for any \( \forall \in Md(Ax^*) \) and \( \varphi \in F^*_\exists, \mathcal{U} \models \varphi[a, x] \) iff \( \mathcal{U} \models \text{Valid}(\Gamma \varphi, \bar{a}, x) \).

**Proof.** See in MENDELSON (1964).

**Theorem 4.13.** (Expressibility.) For any \( Ax^* \) the following properties are expressible in \( Ax^* \) by using the formulas of \( F^*_\exists \):

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 \( \Omega \) be the variable symbol the values of which correspond to the Gödel number of the formula \( \varphi \) that parametrically defines a tree in \( \mathcal{U} \) and let \( \bar{a} \) 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 \( \text{Valid} \) (see Theorem 4.13). Namely \( \text{Valid}(\Omega, \bar{a}, x) \) means that the element \( x \) is an element of the tree \( \Omega \) with parameter \( \bar{a} \) (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

\[
\text{Tree}(\Omega, \bar{a}) = \text{Valid}(\Omega, \bar{a}, \Lambda) \land \forall x (\text{Valid}(\Omega, \bar{a}, x) \rightarrow \text{Valid}(\Omega, \bar{a}, \text{left}(x))).
\]

Let \( \mathcal{U} \in Md(Ax^*) \). Now for fixed \( \Omega, \bar{a} \in A \) take

\[
V_{\Omega, \bar{a}} = \{ v \in A | \mathcal{U} \models \text{Valid}(\Omega, \bar{a}, v) \}.
\]

It is clear that \( V_{\Omega, \bar{a}} \) is a parametrically definable tree in \( \mathcal{U} \). Moreover if \( \varphi \) defines a tree \( V \) by parameters \( \bar{a} \) then, using 4.12, we have \( V = V_{\varphi, \bar{a}} \).
b) Let \( \text{Path} (\Omega, g, \bar{a}, \bar{b}) = \forall x(\text{Valid}(g, \bar{b}, x) \rightarrow \text{Valid}(\Omega, \bar{a}, x)) \land \\
\forall x \forall y \{\text{Valid}(g, \bar{b}, x) \land \text{Valid}(g, \bar{b}, y) \land \text{left}(x) = \text{left}(y) \rightarrow x = y\} \).

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

d) Let \( GF(\Omega_\nu, \Omega_c, \bar{a}) \) be the formula corresponding to property c)

\[
R(\Omega_\nu, \Omega_c, \bar{a}, r, \bar{b}) = GF(\Omega_\nu, \Omega_c, \bar{a}) \land \text{Tree}(r, \bar{b}) \land \forall x[\text{Valid}(r, \bar{b}, x) \rightarrow \\
\text{Valid}(\Omega_v, \bar{a}, x)] \land \forall y \forall z[\text{Valid}(r, \bar{b}, y) \land \text{Valid}(\Omega_v, \bar{b}, z) \land \\
\text{left}(y) = x \land \text{left}(z) = y \land \text{Valid}(\Omega_c, \bar{a}, x) \rightarrow y = z].
\]

It is evident that this formula is good.

g) Let

\[
NL(\Omega_\nu, \Omega_c, \Omega_{\nu_\nu}, \bar{a}) = \exists r \exists \bar{b} (R(\Omega_\nu, \Omega_c, \bar{a}, r, \bar{b}) \land \\
\forall g \forall \bar{c} (\text{Path}(r, g, \bar{b}, \bar{c}) \rightarrow \neg \text{Valid}(\Omega_{\nu_\nu}, \text{pair}(\bar{a}, \bar{c}), g))).
\]

This formula means that there is a run \( r \) with parameters \( \bar{b} \) such that \( r \) is a run of the game defined by \( \Omega_\nu \) and \( \Omega_c \) and no path \( g \) with parameter \( \bar{c} \) of \( r \) belongs to \( \Omega_{\nu_\nu} \).

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 \( \mathcal{U} \) such that for any \( d \in A \) satisfying \( \mathcal{U} \models \zeta^*[d] \) there is a \( d \)-long definable path. Then there is a \( \zeta \)-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 \( \vartheta \)-type nondeterministic programming language \( NP_{\vartheta} \). The programs of the language \( NP_{\vartheta} \) might be "executed" in the models of on arbitrary \( \vartheta \)-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 $U_9$ of 9-type nondeterministic commands consists of the following elements:

(i) $j: y \leftarrow \tau,$
(ii) $j: \square y \equiv \tau,$
(iii) $j: \text{if } \chi \text{ then } \square k_1, ..., k_r$

where $\square \in \{\text{any, every}\}$, $j, k_1, ..., k_r$ are natural numbers, $y \in Y$, $\tau \in T^Y_9$, and $\chi \in Q^Y_9$.

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

Thus the commands of $U_9$ have the form $j: u$, where the natural number $j$ is said to be the *label* of the command.

**Definition 5.2.** A 9-type nondeterministic program $p$ is a nonempty sequence of 9-type commands $p = (i_0: u_0, ..., i_n: u_n) \in U_9^+$ 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 $NP_9$ denote the set of all 9-type nondeterministic programs:

$$NP_9 = \{p | p \text{ is a 9-type nondeterministic program} \}.$$

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

$$
i_0 : u_0
\vdots
i_n : u_n
$$

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

For any nondeterministic program $p = (i_0: u_0, ..., i_n: u_n) \in NP_9$ we use the following notations:

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

$$\text{Var} (j: y \leftarrow \tau) = \text{Var} \tau \cup \{y\},$$
$$\text{Var} (j: \square y \equiv \tau) = \text{Var} \tau \cup \{y\},$$
$$\text{Var} (j: \text{if } \chi \text{ then } \square k_1, ..., k_r) = \text{Var} \chi.$$

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

(ii) $i_{n+1} = \min \{k | k \in \{i_0, ..., i_n\}\}$.

The programming language $NP_9$ 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 $j: \text{if } x=x \then \square k_1, ..., k_r$ is the same as $j: \text{goto } \square k_1, ..., k_r$. We use the latter in the language $NP_9$ 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 9-type system $Ax^\ast$. 
Definition 5.3. Let us fix a system \( Ax^* \). Let us consider an arbitrary model \( \mathfrak{M} \in Md( Ax^* ) \) and an arbitrary program \( p = \langle i_0 : u_0, \ldots, i_n : u_n \rangle \in NP_\xi \).

Let \( q : \text{Var} p \rightarrow A \) be an arbitrary evaluation of the variable symbols of \( p \).

Let \( G_q = ( V_q , C_q ) \) be a game-frame and \( f_q = ( l, s ) : V_q \rightarrow \mathcal{N} \times \text{Var} p \times A \) be a trace with the following properties:

1. \( G_q \) and \( f_q \) are parametrically definable in \( \mathfrak{M} \);
2. (i) \( \mathfrak{M} \models \zeta^* [ \text{length} (v) ] \) for any \( v \in V_q \),
   (ii) \( l(0) = i_0 \) and \( s(0) = q \),
   (iii) if \( l(v) = \{ i_m | m \equiv n \} \) 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 \( S_{V_q}(v) = \{ \text{pair}(v, 0) = w \} \), \( l(w) = i_{m+1} \),
   \( s(w)(x) = \begin{cases} s(v)(x) & \text{if } x \notin \text{Var } p \setminus \{ y \}, \\ t\mathfrak{M}[s(v)] & \text{if } x = y, \end{cases} \)
   (ii) if \( u_m = \Box y \equiv \tau \) then \( S_{V_q}(v) = \{ \text{pair}(v, e) | 0 \equiv e \equiv \tau \mathfrak{M}[s(v)] \} \) and for any \( w \in S_{V_q} \), \( l(w) = i_{m+1} \),
   \( s(w)(x) = \begin{cases} s(v)(x) & \text{if } x \notin \text{Var } p \setminus \{ y \}, \\ \text{right}(w) & \text{if } x = y, \end{cases} \)
   (iii) if \( u_m = \text{if } \chi \text{ then } \Box k_1, \ldots, k_r \) then
     a) if \( \mathfrak{M} \models \neg \chi [s(v)] \) then \( S_{V_q}(v) = \{ \text{pair}(v, 0) \} \) and \( l(w) = i_{m+1} \) for \( w \in S_{V_q}(0) \),
     b) if \( \mathfrak{M} \models \chi [s(v)] \) then \( S_{V_q}(v) = \{ \text{pair}(v, k_j) | j \in [1, r] \} \) and \( l(w) = \text{right}(w) \) 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

\[ C_q = \{ v \in V_q | l(v) = i_m \text{ and } u_m \text{ is either any } y \equiv \tau \text{ or if } \chi \text{ then any } k_1, \ldots, k_r \}. \]

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 \( \mathfrak{M} \) 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 \( \mathfrak{M} \)-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 \textit{stop} is not included in \( U_q \).

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 \textit{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 \( \Box \), the game-tree branches out according to the possible choices. The cases \textit{any} and \textit{every} differ from one another simply whether player \( A \) or \( B \) moves. The correctness of the above definition is ensured by the following

\textbf{Lemma 5.4.} For an arbitrary program \( p \in NP_A \), \( q \in Md(Ax^{*}) \) and \( q^r : \text{Var} p \rightarrow A \) there is a unique trace of \( p \) in \( \mathcal{U} \) 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).

\textbf{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 \( v \notin \mathcal{A} \) so \( w=\text{left}(v) \in V \) 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 \( \mathcal{U} \) starting with \( q \).

\begin{align*}
q(v, l, y_1, \ldots, y_k, a_1, \ldots, a_k) & \leftrightarrow (l = i_0 \land v = A \land \bigwedge_{j=1}^{k} y_j = a_j) \\
\forall v^*, l^*, y_1^*, \ldots, y_k^* \{q(v^*, l^*, y_1^*, \ldots, y_k^*, a_1, \ldots, a_k) \land l^* \in \{i_m | m \leq n\} \land \xi^*(\text{length } v) \land \\
\bigwedge_{m=0}^{n} l^* = i_m & \rightarrow \left\{ v = (v^*, 0) \land \bigwedge_{i \neq s}^{k} y_i^* = y_i \land y_s = \tau[y^*/y] \land l = i_{m+1} \right\} \land \\
\bigwedge_{m=0}^{n} l^* = i_m & \rightarrow \left\{ v = (v^*, y_2) \land \bigwedge_{i \neq s}^{k} y_i^* = y_i \land y_s = \tau[y^*/y] \land l = i_{m+1} \right\} \land \\
\bigwedge_{m=0}^{n} l^* = i_m & \rightarrow \left\{ \bigwedge_{i=1}^{k} y_i^* = y_i \land (\forall \chi \rightarrow v = (v^*, 0) \land l = i_{m+1}) \land \\
\bigwedge_{m=0}^{n} l^* = i_m & \rightarrow \left\{ \bigwedge_{i=1}^{r} y_i^* = y_i \land (\forall \chi \rightarrow v = (v^*, k_i) \land l = k_i) \right\} \right\}.
\end{align*}
Applying Theorem 3.15 this equation has a minimal solution say \( q^* \in F_q \).

It is evident that

\[
\Omega_v(w, a) = \exists l, y_1, \ldots, y_k q^*(w, l, y_1, \ldots, y_k, a_1, \ldots, a_k)
\]

defines a tree in \( \mathcal{U} \) and if

\[
\Omega_c(w, a) = \exists l, y_1, \ldots, y_k \left[ q^*(v, l, y_1, \ldots, y_k, a_1, \ldots, a_k) \wedge \bigvee_{i=0}^{n} \bigvee_{m=0}^{u_m} \bigvee_{m=0}^{u_m} \bigvee_{m=0}^{u_m} \bigvee_{m=0}^{u_m} \right]
\]

ten (\( \Omega_v, \Omega_c \)) defines the associated \( q \)-game-frame \( GF_q \) and \( q^* \) parametrically defines the trace of \( p \) in \( \mathcal{U} \) starting with \( q \).

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

6. Descriptive language

Let \( Ax^* \) be a \( \vartheta \)-type system and \( \mathcal{U} \in Md(Ax^*) \) an arbitrary model. Let \( p = (i_0: u_0, \ldots, i_n: u_n) \in NP_\vartheta \) and \( q: \text{Var} \rightarrow A \) 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 \( \Gamma_B \) 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 \( \Gamma_B \) 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 \( \Gamma_B \).

A. Let \( \Gamma_B \) be the set of those definable paths of the game-frame \( GF_q \) in which there exists a situation (node) \( v \) such that \( l(v) \in \{ i_m | m \equiv n \} \), i.e. \( \Gamma_B \) 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 \( \zeta \)-long). \( \square \)

B. Let \( \psi \in F^A_x^* \) be an arbitrary formula that is called the output condition and let \( \Gamma_B \) be the set of those definable paths of \( V_q \) each of that contains a node \( v \) such that

(i) \( l(v) \notin \{ i_m | m \equiv n \} \) and

(ii) \( \mathcal{U} \not\models \psi[s(v)] \).

So \( \Gamma_B \) consists of those executions of the program \( p \) that terminate without satisfying the output condition \( \psi \).

In this case the non-losing strategy of \( A \) means the partial correctness of the program \( p \) with respect to the output condition \( \psi \) (while the input condition is any tautology e.g. \( x = x \)). \( \square \)
C. Let \( \varphi, \psi \in F_{\mathcal{B}}^q \) be arbitrary formulas that are called input and output conditions respectively and let \( \Gamma_B \) be the set of those definable paths of \( V_q \) for each of which:

(i) \( \mathcal{U} \models \varphi[s(0)] \),

(ii) there exists a node (in the path) such that \( l(v) \in \{i_m | m \equiv n\} \) and \( \mathcal{U} \not\models \psi[s(v)] \).

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

D. Let \( \varphi, \psi \in F_{\mathcal{B}}^q \) be arbitrary formulas and let \( \Gamma_B \) be the set of definable paths of \( V_q \) such that each satisfies both

(i) \( \mathcal{U} \models \varphi[s(0)] \) and

(ii) either it does not terminate or it has a node \( v \) such that \( l(v) \in \{i_m | m \equiv n\} \) and \( \mathcal{U} \not\models \psi[s(v)] \).

Now the non-losing strategy of \( A \) means the total correctness of the program \( p \) w.r.t. the input condition \( \varphi \) and output condition \( \psi \). ■

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 \( X_p \subseteq Y \) of variable symbols that duplicates the variable symbols \( Y_p \) and refers to the input values of the latter.

If \( Y_p = \{y_1, ..., y_k\} \) then \( X_p = \{x_1, ..., x_k\} \) 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 \( \varphi \in F_{\mathcal{B}}^q \) is said to be an input condition for a fixed program \( p \) iff \( \text{Var} \varphi \subseteq X_p \). Moreover, a formula \( \psi \in F_{\mathcal{B}}^q \) is called output condition for the program \( p \) iff \( \text{Var} \psi \subseteq X_p \cup Y_p \cup \{t\} \).

Now let us consider the further cases.

E. Let \( \varphi \) and \( \psi \) be input and output conditions for a fixed program \( p \) respectively and let \( \Gamma_B \) be the set of those definable paths of \( V_q \) which satisfy \( \mathcal{U} \models \varphi[q] \) and have a node \( v \) such that \( l(v) \in \{i_m | m \equiv n\} \) and \( \mathcal{U} \not\models \psi[q, s(v), \text{length}(v)] \).

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 with the strategy is such that it starts with input \( q \) that satisfies condition \( \varphi \) and when it stops it satisfies condition \( \psi \). ■

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

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 \( \varphi \) and it terminates by satisfying condition \( \psi \). ■
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 $\Gamma_B$, 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 $\Gamma_B$ 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.** $F_d = \{(\varphi, p, \psi) | p \in NP, \varphi$ and $\psi$ are input and output conditions for $p$ respectively$\} \cup \{(\varphi, p, \psi) | p \in NP, \varphi$ and $\psi$ are input and output conditions for $p$ respectively$\}$. \(
\)

When we are interested only in the triple of $\varphi, p, \psi$ 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 $\chi = [\varphi, p, \psi] \epsilon F_d$ and let $\mathcal{A} \in Md(Ax^*)$. According to Lemma 5.4 for any $q: \text{Var} \rightarrow A$ there is a game-frame $G_q = (V_q, C_q)$ and a function $f_q = (l, s): V_q \rightarrow A$ and they are definable in $Ax^*$. Starting out from the game-frame $G_q$ and taking the formula $\chi$ into account the following game can be constructed:

(i) $G_A^\chi = \emptyset$,

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

a) First if $\chi = (\varphi, p, \psi)$ then $G^\chi = \{\gamma$ is a definable path in $V_q |$ there is a $v \in \gamma$ such that $l(v) \epsilon \{i_m | m \leq n\}$ and $\mathcal{A} \not\models \psi[q, s(v), \text{length}(v)]\}.$

b) Secondly if $\chi = (\varphi, p, \psi)$ then let $G^\chi = \{\gamma$ is a definable path in $V_q |$ $\gamma$ is either infinite or there is a $v \in \gamma$ such that $l(v) \epsilon \{i_m | m \leq n\}$ and $\mathcal{A} \not\models \psi[q, s(v), \text{length}(v)]\}.$

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

By using the technique used in the proof of Theorem 4.12 it is not difficult to verify that the set $\Gamma^\chi$ 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 $\mathcal{A} \in Md(Ax^*)$ be arbitrary, $\chi \epsilon F_d$ and $q: X_a \rightarrow A$. Then the associated q-game $(V_q, C_q, \emptyset, \Gamma^\chi)$ is parametrically definable in $Ax^*$. \(\square\)

Now we turn to the definition of the semantics of the descriptive language to be defined.
Definition 6.3. Let $\mathcal{U} \in Md(Ax^*)$ and $\chi \in F_\mathfrak{I}$ be arbitrary. $\mathcal{U} \models \chi$ iff for any $q : \text{Var} p : A$ player $A$ has a non-losing definable strategy in the associated $q$-game $G_q^*$. \hfill \square

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

Definition 6.4. For an arbitrary $\mathfrak{I}$-type system $Ax^*$ the descriptive language expressing partial and total correctness and describing the corresponding time condition is

$$D_{Ax^*}^d = (F_{\mathfrak{I}}, Md(Ax^*), \models).$$

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

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 $\chi \in F_\mathfrak{I}$ there is a formula $\chi^* \in F_{\mathfrak{I}}^X$ such that for any $\mathcal{U} \in Md(Ax^*)$, $\mathcal{U} \models \chi$ iff $\mathcal{U} \models \chi^*$.

The function rendering $\chi^*$ to $\chi$ is recursive.

Moreover, there is a formula $\sigma_{\chi}(\alpha, \nu, w) \in F_{\mathfrak{I}}^X$ such that if $\mathcal{U} \models \chi$ then for any $q : \text{Var} p : A$, $\sigma_{\chi}[q]$ defines a non-losing strategy of player $A$ in the associated $q$-game $G_q^*$.

Proof. The existence of the formula $\chi^*$ for a given $\chi \in F_\mathfrak{I}$ is an easy consequence of Theorem 4.9. The recursiveness of the function that renders the corresponding $\chi^*$ to each of them can be established by using Gödel-numbering. We omit the details. \hfill \square

Let $E_{Ax^*}$ denote the set of all formulas that are true in the models of $Ax^*$, i.e.

$$E_{Ax^*}^d = \{ \chi : Ax^* \models \chi \}.$$  

From the point of view of the usability of the descriptive language $D_{Ax^*}^d$ for the theory of programming the handlebility of $E_{Ax^*}$ 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 $E_{Ax^*}^d$, i.e. the descriptive language $D_{Ax^*}^d$ is complete.

Proof. Immediate from 6.6. \hfill \square

Remark 6.8. If $\mathcal{U} \models \chi$ would be defined so that for any input data $q$ in the game $G_q^*$ player $A$ had a non-losing strategy (without the assumption of definability) then the language $D_{Ax^*}^d$ would lose the completeness, even more, it would be neither $\omega$-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 \( D_{\text{class}} \) 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 \( F_3 \) can be coded in the syntax \( F_{9}^{*} \) 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 \( \text{Lab}: NP_{9} \rightarrow P(\mathbb{N}) \) be the function rendering to each non-deterministic program \( p=(i_0: u_0, \ldots, i_n: u_n) \in NP_{9} \) the set of labels occurring in it as follows.

\[
\text{Lab}(p) = \{m \mid m \leq n+1 \} \cup \{k \mid \text{there is an } m \text{ such that } u_m = \text{if } x \text{ then } \square \ldots k \ldots \square \in \{\text{any, every}\}\}.
\]

**Definition 7.2.** Let us fix a \( \vartheta \)-type system \( Ax^{*} \) and let \( \varphi = \{p, \psi\} \) be an arbitrary formula of \( F_{9}^{*} \). Let \( \Phi: \text{Lab}(p) \rightarrow F_{9}^{*} \) be a function which to each label of the program \( p \) renders a formula \( \Phi(i) \) with variable symbols \( x, y, y \). Remember that \( x \) serves to duplicate the program variable \( y \) and \( t \) refers to time. Further on we write \( \Phi_i \) instead of \( \Phi(i) \). The function \( \Phi \) is said to be the description of \( \varphi \) w.r.t. \( Ax^{*} \) iff it satisfies the following conditions:

1. \( Ax^{*} \vdash \varphi \rightarrow \Phi_{i_0}[0/t, \vec{x}/\vec{y}] \)  
   (input rule)

2. if \( i_m: y = \tau \) then
   \[
   Ax^{*} \vdash \Phi_{i_m} \rightarrow \Phi_{i_{m+1}}[\tau/y, t+1/t] 
   \]
   (assignment rule)

3. if \( i_m: \text{any } y \equiv \tau \) then
   \[
   Ax^{*} \vdash \Phi_{i_m} \rightarrow \exists z (z \equiv \tau \land \Phi_{i_{m+1}}[z/y, t+1/t])
   \]
   (assignment rule of A’s choice)

4. if \( i_m: \text{every } y \equiv \tau \) then
   \[
   Ax^{*} \vdash \Phi_{i_m} \rightarrow \forall z (z \equiv \tau \land \Phi_{i_{m+1}}[z/y, t+1/t])
   \]
   (assignment rule of B’s choice)

5. if \( i_m: \text{if } x \text{ then any } k_1, \ldots, k_r \) then
   \[
   Ax^{*} \vdash \Phi_{i_m} \land \forall x \rightarrow \Phi_{i_{m+1}}[t+1/t], \quad Ax^{*} \vdash \Phi_{i_m} \land x \rightarrow \bigvee_{j=1}^{r} \Phi_{k_j}[t+1/t]
   \]
   (rule of conditional jump of A’s choice)

6. if \( i_m: \text{if } x \text{ then every } k_1, \ldots, k_r \) then
   \[
   Ax^{*} \vdash \Phi_{i_m} \land \forall x \rightarrow \Phi_{i_{m+1}}[t+1/t], \quad Ax^{*} \vdash \Phi_{i_m} \land x \rightarrow \bigvee_{j=1}^{r} \Phi_{k_j}[t+1/t]
   \]
   (rule of conditional jump of B’s choice)

7. \( Ax^{*} \vdash \Phi_{\psi} \)  
   (output rule for \( z \notin \{i_m \mid m \leq n\} \))

5 Acta Cybernetica
Moreover if $\chi=[\varphi, p, \psi]$ then
(viii) for any $z \in \{1, \ldots, n\}$
\[ Ax^* \vdash \exists v \forall \eta [v_1 \rightarrow v \equiv t] \] (rule of termination).
We denote by $Ax^* \vdash \chi$ the fact that the formula $\chi$ has a description w.r.t. $Ax^*$.

Now let us see some remarks on the above definition:
Each formula $\Phi_i (0 \leq i \leq n)$ shows the conditions the data should satisfy before executing the command labelled by $i$ and the corresponding time conditions. We note that $z$ can be used in any formula $\Phi_z$.

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 $t$ 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 $y$. 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 $\chi$ 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 $\text{stop} = \infty$. 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 $\Phi^*_z \doteq \exists x \Phi_z$ for each descriptive formula $\Phi_z$.

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 $\mathfrak{q}$-type system $Ax^*$ and $\chi \in F_\mathfrak{q}$, $Ax^* \vdash \chi$ iff $Ax^* \vdash \chi$.

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

Let us fix a $\chi=[\varphi, p, \psi]$ and let $\Phi: \text{Lab} p \rightarrow F_\mathfrak{q}^*$ be a description of $\chi$ w.r.t. $Ax^*$. Let $\mathfrak{q} \in \text{Md}(Ax^*)$ and $q: \text{Var} p \rightarrow A$ be arbitrary. We must prove that in the $q$-game $G_q^*=(V_\mathfrak{q}, C_\mathfrak{q}, \emptyset, \Gamma_\mathfrak{q})$ player $A$ has a non-losing strategy.

Let $(l, s): V_q \rightarrow N \times \text{Var} p A$ be the trace of $p$ in $\mathfrak{q}$ 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 $(v)=i_m$ and $u_m=\text{any } y \equiv \tau$ or $u_m=\text{if } x \text{ then } k_1, \ldots, k_r$. If $\mathfrak{q} \vdash \Phi_{l(v)}[q, s(v), \text{length } v]$ then let $\text{str } (v)$ be the minimal element of $S_{y_q}(v)$. (It might be arbitrary but we have to be
sure that \textit{str} is parametrically definable.) So let us suppose that \( \mathcal{U} \models \Phi_{i_m}[q, s(v), \text{length} (v)] \). First let \( u_m=\text{any } y \equiv r \). By our assumption with \( i_m=l(v) \)
\[
\mathcal{U} \models \Phi_{i_m}[q, s(v), \text{length} (v)]
\]
and thus
\[
\mathcal{U} \models \exists z \,(z \equiv r \land \Phi_{i_m+1}[r/y, t+1/t][q, s(v), \text{length} (v)]).
\]
Now let \( z^* \) be the minimal \( z \in A \) such that
\[
\mathcal{U} \models z \equiv r \land \Phi_{i_m+1}[r/y, t+1/t][q, s(v), \text{length} (v), z].
\]
It is clear that \( \text{pair} (v, z^*) \in S_{\nu_q} (v) \) and so let
\[
\text{str} (v) \overset{d}{=} \text{pair} (v, z^*).
\]
Obviously
\[
\mathcal{U} \models \Phi_{i_m+1}[q, s(\text{str} (v)), \text{length} (\text{str} (v))]
\]
In the end let \( u_m=\text{if } x \text{ then any } k_1, \ldots, k_r \). Since \( v \in C_q \) so \( \mathcal{U} \models x[s(v)] \). Using
\[
\mathcal{U} \models \Phi_{i_m} \land x \rightarrow \bigvee_{j=1}^r \Phi_{k_j}[t+1/t] \text{ and } \mathcal{U} \models \Phi_{i_m} \land x[q, s(v), \text{length} (v)]
\]
we have
\[
\mathcal{U} \models \bigvee_{j=1}^r \Phi_{k_j}[t+1/t][q, s(v), \text{length} (v)].
\]
Let \( k^* \) be the minimal \( k \in \{k_1, \ldots, k_r\} \) for which
\[
\mathcal{U} \models \Phi_k[t+1/t][q, s(v), \text{length} (v)]
\]
and take \( \text{str} (v) \overset{d}{=} \text{pair} (v, k^*) \).

Thus the strategy \textit{str} is defined. Since \( G_q \) is definable and the method used in the above can be defined in \( F_q^A \), so \textit{str} is also parametrically definable. The fact, that \textit{str} is a non-losing strategy follows from the following

\textbf{Lemma 7.4.} Let \( \pi \) be a parametrically definable path in the run \( R_{str} \) of the strategy \textit{str}. For every \( v \in \pi \)
\[
\mathcal{U} \models \Phi_{l(v)}[q, s(v), \text{length} (v)].
\]
In fact if \( l(v) \notin \{i_m | m \equiv n\} \) then
\[
\mathcal{U} \models \psi [q, s(v), \text{length} (v)].
\]
Moreover, if \( \chi=\varphi, p, \psi \) then \( \pi \) terminates.

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

If \( \chi=\varphi, p, \psi \) then let us apply (viii) of Definition 7.2 and we get a \( t \in A \) such that if \( \text{length} (v) \rightarrow t \) then for any \( z \in \{i_m | m \equiv n\} \), \( \mathcal{U} \models \Phi_z [q, s(v), \text{length} (v)] \). Hence using \( \mathcal{U} \models \Phi_{l(v)}[q, s(v), \text{length} (v)] \) we have \( l(v) \notin \{i_m | m \equiv n\} \).

The proof of the lemma is completed. \( \square \)
Now the proof of the theorem is continued by showing that if $Ax^* \equiv \chi$ then $Ax^* \vdash \chi$.

Let $\chi=\varphi, p, \psi$ be such that $Ax^* \equiv \chi$. Let $\Omega_V(\bar{a}, v)$ and $\Omega_C(\bar{a}, w)$ be the formulas which parametrically define the game-frame $GF=(V, C)$ in $Ax^*$ generated by $\chi$. By the second part of Theorem 6.6 there is a formula $\sigma_\chi(\bar{a}, v, w)$ with the same parameters as $\Omega_V, \Omega_C$ which in $Ax^*$ parametrically defines a non-losing strategy for player $A$ in the game $G^*_\chi$. Hence there is a formula $\varphi_\chi(\bar{a}, r) \in F^*_\chi$, which defines in $Ax^*$ the run of the strategy defined by $\sigma_\chi$. Now we give a description of $\chi$ in $Ax^*$:

$$\Phi_\chi(\bar{x}, \bar{y}, t) \vdash \varphi(\bar{x}) \land \exists v \{\varphi_v(\bar{x}, v) \land l(v) = z \land \bar{y} = s(v) \land \text{length}(v) = t\}.$$

Since $s$ is parametrically definable in $Ax^*$ by using only the parameter $\bar{x}$ so $\Phi_\chi \in F^*_\chi$. It is clear that $\text{Var} \Phi \subset \{x_1, \ldots, x_k, y_1, \ldots, y_k, t\}$.

The following lemma is enough to complete the proof.

**Lemma 7.5.** The above function $\Phi$ is a description of $\chi$ w.r.t. $Ax^*$.

**Proof.** Let $\mathcal{U} \in Md(Ax^*)$ and $q: \text{Var} p \rightarrow A$ be arbitrary. We have to prove that for any $\bar{y}, t \in A$ the rules in Definition 7.2 holld. (i)–(vii) can be proved in the same way so e.g. we prove (iii). Let $\bar{y}, t \in A$ be such that $\mathcal{U} \models \Phi_{im}[q, \bar{y}, t]$ and $u_m = \text{every } y \subseteq t$. Let $G^*_\chi=(V_q, C_q, 0, \Gamma^*_q)$ be the associated $q$-game generated by $\chi$ and let $(l, s)$ be the trace of $p$ in $\mathcal{U}$. By the definition of $\Phi_{im}$ there is a $v \in V_q$ such that $\bar{y} = s(v)$ and $t = \text{length}(v)$. Moreover $v \in R_{str}$ where $str$ is the non-losing strategy of player $A$ defined by $\sigma$ and $q$. Let $z \equiv \tau_{\mathcal{U}}[s(v)]$ be arbitrary. Then $w = \text{pair}(v, \bar{z}) \in R_{str}$, $\text{length}(w) = t+1$,

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

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

This means that

$$\mathcal{U} \models \Phi_{im}[q, s(w), l(w)].$$

Hence

$$\mathcal{U} \models \Phi_{im+1}[z/y, t+1/t][q, \bar{y}, t].$$

Summarizing:

$$\mathcal{U} \models \Phi_{im} \rightarrow \forall z (z \subseteq t \land \Phi_{im+1}[z/y, t+1/t]).$$

If $\chi=\varphi, p, \psi$ we must prove that for any $z \notin \{i_m|m \equiv n\}$,

$$\mathcal{U} \models \exists T \forall \bar{y} \forall t [\Phi_{\chi}(\bar{y}, t) \rightarrow t \equiv T].$$

By the definition of $G^*_\chi$ 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 $T \in A$ such that for any $v \in R_{str}$, $\text{length}(v) \equiv T$. As in the first part of the proof for any $(\bar{y}, t) \in A$ with

$$\mathcal{U} \models \Phi_{\chi}[q, \bar{y}, t]$$

there is a $v \in R_{str}$ such that $s(v) = \bar{y}$ and $\text{length}(v) = t$. Hence $t \equiv T$ and thus

$$\mathcal{U} \models \forall \bar{y} \forall t [\Phi_{\chi}(\bar{y}, t) \rightarrow t \equiv T][q, T].$$

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:

\[
P_1 =
\]

0: any \( y_1 \equiv \max(a, b) \)

1: if \( a \mid y_2 \land b \mid y_2 \land y_2 \neq 0 \) then 5

2: if \( y_2 > ab \) then 5

3: \( y_2 = y_2 + y_1 \)

4: goto 1

The input and output conditions are respectively

\[
\varphi = a > 0 \land b > 0 \land y_2 = 0
\]

and

\[
\psi = y_2 = [a, b].
\]

Let \( Ax \) be the set of axioms, which contains \( PA \) and the axiomatic definition of \( |, \max, \prec, \equiv, \) and \( \ldots, \ldots \). Let

\[
Ax^* = Ax \cup \{ \forall x \xi(x) \}.
\]

We are going to prove that

\[
Ax^* \vdash (\varphi, p_1, \psi).
\]

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 \( \chi \) w.r.t. \( Ax^* \):

\[
\begin{align*}
\Phi_0 &= a > 0 \land b > 0 \land y_2 = 0 \\
\Phi_1 &= y_1 > 0 \land (y_1 \mid a \lor y_1 \mid b) \land \forall z (z \equiv y_2 \land y_2 \mid z \leftarrow \forall z a \lor \forall z b) \\
\Phi_2 &= y_2 > 0 \land (y_1 \mid a \lor y_1 \mid b) \\
\Phi_3 &= \Phi_0 \land y_2 \equiv ab \\
\Phi_4 &= \Phi_1 \\
\Phi_5 &= y_2 = [a, b]
\end{align*}
\]

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

It is obvious that the greater value is chosen for \( y_1 \) from that of satisfying the condition \( \Phi_1 \) the sooner termination of the program occurs. That is why from
the output condition we claim the expression of early termination. So let

\[ \psi = y_2 = [a, b] \land t \leq \frac{4([a, b] + 1)}{\text{max}(a, b)}. \]

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

- $\bar{\tau}_0 \overset{d}{=} a > 0 \land b > 0 \land y_2 = 0$
- $\bar{\tau}_1 \overset{d}{=} y_1 = \max(a, b) \land y_2 = \frac{t-1}{4} y_1 \land y_2 \equiv ab \land \forall z (z \equiv y_2 \land y_1 \land -\exists z|a \lor \exists z|b)$
- $\bar{\tau}_2 \overset{d}{=} y_1 = \max(a, b) \land y_2 = \frac{t-2}{4} y_1 \land y_2 \equiv ab \land \forall z (z \equiv y_2 \land y_1 \land -\exists z|a \lor \exists z|b)$
- $\bar{\tau}_3 \overset{d}{=} \bar{\tau}_2[t-3/t]$
- $\bar{\tau}_4 \overset{d}{=} \bar{\tau}_1[t+1/t]$
- $\bar{\tau}_5 \overset{d}{=} y_2 = [a, b] \land t \leq \frac{4([a, b] + 1)}{\text{max}(a, b)}.$

From the descriptions $\Phi$ and $\bar{\tau}$ 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 occurrence of *any* we get a program $p'_1$ such that

\[ Ax^* \equiv [\varphi, p'_1, t \leq 4ab + 4]. \]

### 8.2 Pattern matching

Now we show how the nondeterministic programming language $NP_9$ can be used for handling the problem of pattern matching. First let us specify what pattern-matching is.

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

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 $\langle X(j) \rangle_{j \equiv n}$.

Since vectors and matrices are not allowed in our language $NP_9$ explicitly we have to suppose that the type $\mathcal{G}$ contains the function symbols $vcomp$ and $mcomp$, for which $\mathcal{G}(vcomp) = 2$ and $\mathcal{G}(mcomp) = 3$. The function symbol $vcomp$ provides the $i$-th component of the vector having the code $X$. Analogously $mcomp$ provides the $(i, j)$-th component of the matrix having the code $A$. 
Let us consider the following program the input data of which are \( A, x, n, m \).

\[
p_2 \overset{d}{=} \\
0: \text{any } i \leq m \\
1: \text{every } j \leq n \\
2: \text{if } vcomp(x, j) \leq mcomp(A, i, j) \text{ then } 5 \\
3: u = 0 \\
4: \text{goto } 6 \\
5: u = 1
\]

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 \( A, x, n, m \) is correct iff such a row \( s \) can be chosen for the label 0 that can be matched to \( X \).

### 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 \( k + 1 \mid n \).

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

\[
p_3 \overset{d}{=} \\
0: \text{any } y \leq k \\
1: \text{if } y = 0 \text{ then } 9 \\
2: \text{if } y \leq n \text{ then } 11 \\
3: n \leftarrow n - y \\
4: \text{every } y \leq k \\
5: \text{if } y = 0 \text{ then } 11 \\
6: \text{if } y \leq n \text{ then } 9 \\
7: n \leftarrow n - y \\
8: \text{goto } 0 \\
9: u = 0 \\
10: \text{goto } 12 \\
11: u = 1
\]
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 $i$: $\square y \equiv t$, $\square \in \{\text{any, every}\}$. However we could have given them in the following form $i$: $\square \tau_1 \equiv y \equiv \tau_2$ or $i$: $\square y \neq \varnothing$ (where $\tau_1, \tau_2 \in T_3$ and $\varnothing \in F_3$ 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 $\psi^d = u = 1$. Since in the game $G$ draw 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 $\chi$ w.r.t. $Ax^*$. It is easy to verify that the formulas below define a description of $\chi$ w.r.t. $Ax^*$:

$$\begin{align*}
\Phi_0^d &= \neg k + 1|n \\
\Phi_1^d &= 1 \equiv y \equiv k \land y \equiv n \mod (k + 1) \\
\Phi_2^d &= 1 \equiv y \equiv k \land y \equiv n \mod (k + 1) \\
\Phi_3^d &= y < n \land y \equiv n \mod (k + 1) \\
\Phi_4^d &= k + 1|n \land n > 0 \\
\Phi_5^d &= y \equiv k \land n > 0 \land k + 1|n \\
\Phi_6^d &= 1 \equiv y \equiv k \land k + 1|n \land n > 0 \\
\Phi_7^d &= 1 \equiv y \equiv k \land k + 1|n \land n > 0 \\
\Phi_8^d &= \neg k + 1|n \\
\Phi_9^d &= \text{false} \\
\Phi_{10}^d &= \text{false} \\
\Phi_{11}^d &= \text{true} \\
\Phi_{12}^d &= u = 1.
\end{align*}$$
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.

References


(Received Sept. 7, 1979)