BANACH CENTER PUBLICATIONS
VOLUME 2

THEORY OF COMPUTING SYSTEMS

W. BARTOL, Z. RAŚ, A. SKOWRON

Institute of Mathematics, University of Warsaw, Warsaw

CONTENTS

INTRODUCTION

At the present state of Computer Science it seems that a mathematical theory of computing systems needs some motivation. Computer Science being a relatively new field of scientific interest, it has achieved many important, often very deep results, e.g., in automata theory, mathematical linguistics, theory of complexity, theory of programs, etc. All these theories refer in a more or less direct way to some physical reality—which is the computer. However, it is a paradox that in the whole Computer Science there is no widely agreed formal notion of a computer (as well as of a programming language, for instance). It seems that an explanation of this fact is to be sought in the historical background: the construction of computers created so many practical or specific problems, related both to software and hardware, that the main effort of computer scientists has been concentrated on those problems and on particular branches inspired by them, and not on a general theory of computers. Moreover, the constant development of computer technology makes it difficult to grasp the essential features of a computer and to describe them formally.

Nevertheless, the rapid progress in different fields of Computer Science creates a need for a common basis—both linguistic and theoretical—which would enable us to speak of all the problems related to computers within a single coherent theory. It seems natural that such a theory should have among its basic notions those of a computer and of a program and that an exact understanding of these notions and of their relationship should be the initial point for the whole field. Mathematics offers both the flexibility of language and the precision of notions needed to develop such a theory.

The idea of introducing and investigating a mathematical definition of a computer is not a very new one. It appeared in the sixties in papers by Elgot and Robinson, Kalmar, Wagner and a few others. Also in recent years new definitions of computers have been introduced by various authors (for example, by Amoroso and Bloom). However, none of them has served as yet as a basis for some wider investigations.

In the present paper we give a mathematical description of a computer on two levels: a very general model and a more structured one, which is closer to von Neumann type computers. Both models are due to Pawlak ([19], [21]). The more general model, called an iterative system, is dealt with in Chapter I. The set of computations of an iterative system is characterized and the effect of boolean operations on sets of computations is examined. Homomorphisms are introduced and their relation to the problem of simulation is investigated. Congruences in iterative systems (and their lattices) are described and used to decompose systems into a subdirect product of simpler systems, on the basis of an analogue of Birkhoff's Theorem. A more specific approach to simulation is presented, where simulation preserves some distinguished properties.

In Chapter II we introduce a special type of iterative systems, in which the set of states and the transition function are described in much more detail, thus making of this model a better counterpart of a real computer. The notion of a program is also introduced and investigated. Using these notions, we consider such problems as the reduction of an instruction list of a computer, decomposition of programs and their simplification and the classification of programs according to their algebraic or computational complexity.

All the theorems, lemmas and corollaries are doubly numbered: by the section number and the consecutive number within the section, and references within the same chapter are made through these numbers. References to another chapter are made by adding the chapter number, e.g., Theorem I.3.2 means the second theorem in Section 3 of Chapter I.

The results contained in this paper are based on the proceedings of a seminar on the theory of computing systems, which was held at the Stefan Banach International Mathematical Center in Warsaw during the Semester on Mathematical Foundations of Computer Science, February–June 1974. However, some results have been revised and some are new.

BASIC NOTATION

For any relation $\varrho \subset X \times Y$ let $\mathcal{D}(\varrho)$ be the domain of ϱ and $R(\varrho)$ the counter-domain of ϱ. In particular, for any partial function $f\colon X \to Y$, $D(f)$ will be the domain and $R(f)$ the range of f. When $D(f) = X$, we shall call f a *function* (or a *total function*). The set of all partial functions from X into Y will be denoted by $Y^{[X]}$, while Y^X denotes as usual the subset of all total functions.

If $A \subset X$, we shall denote by $f|A$ the restriction of f to the set A, i.e. $f|A = f \cap A \times Y$. However, if f is a sequence over X and $A \subset X$, then $f|A$ is the subsequence of f consisting of all elements of f which are in A.

The number of arguments of a (partial) function f will be denoted by $\mathrm{ar}(f)$.

Let $f\colon X \to Y$ be a partial function. Then f^0 is a total identity function in X; for $n \geqslant 0$, $f^{n+1}(x)$ will be defined iff $f^n(x)$ is defined and $f^n(x) \in D(f)$: if $x \in D(f^{n+1})$, then $f^{n+1}(x) = f(f^n(x))$.

The image of $A \subset X$ under f will be denoted by $f(A)$ and the inverse image of $B \subset Y$ by $f^{-1}(B)$ for any partial function $f\colon X \to Y$. We shall often identify one-element sets with their elements, thus, e.g., $f^{-1}(b)$ will be the inverse image of $\{b\}$.

The set of all non-negative integers will always be denoted by N and the set of all integers by Z. For any set X, card (X) is the cardinal number of X. If $\{X_i\}_{i \in I}$ is a family of sets, we shall denote its product by $\prod_{i \in I} X_i$ (or $X_1 \times \ldots \times X_n$, when $I = \{1, \ldots, n\}$).

The set of all non-empty finite sequences over X will be denoted by X^+, the set of all finite sequences over X will be denoted by X^*, while X^∞ will be the set of all non-empty sequences—finite or infinite—over X. If $d \in X^+$, we shall denote its length by $l(d)$ and we shall write $l(d) = \omega$ when d is an infinite sequence.

If R is an equivalence relation in X, we shall denote by $[x]_R$ the equivalence class of x (and we shall write $[x]$ when the relation is understood).

An ordering relation is a relation which is reflexive, antisymmetric and transitive.

We assume that all the sets considered here are subsets of a certain universe. Thus the class of all sets satisfying some condition is again a set.

The end of a proof or an example will be marked by ■.

I. ITERATIVE SYSTEMS

In the present chapter we shall consider a very general mathematical model of a computer. The model being a simple partial algebra, most of its properties may be described algebraically, e.g., through homomorphisms or congruences. It is also possible to speak of simulation of such systems and this can be done in more than one way, depending on what is expected from the simulation. Properties of these algebras may be used to describe more structured models, which will be treated in the next chapter.

Considering any real digital computer, it may be most generally visualized as an object with a memory and a control acting upon the former. In fact, memory would mean here everything that determines the state of a computer at some given moment, i.e. not only the "true" memory where information is stored, but also the state of input/output devices, some internal states of the computer and so on. In this way a computer may be described by two elements: a set X, corresponding to the set of memory states in a computer and a function π (partial, in general), acting on memory states and corresponding to the computer control. Any such pair will be called a *functional iterative system* (or simply, an *iterative system*). Thus, an iterative system $\langle X, \pi\rangle$ is a partial unary algebra with one operation π in the set X. We shall call X the *set of states* of the system and π will be called the *transition function*.

Such a general model of a computer is certain to have its faults and merits. The latter seem to be: its ability to describe those properties of computers which are universal, i.e. do not depend on any particular physical realizations of computing machines; the possibility that it offers of considering on a uniform basis different objects which may be described as iterative systems, e.g. computers, but also automata, grammars or programs; finally, last but not least, its being easily manageable, since it is a rather simple mathematical object.

As often happens, what is a merit in one approach may turn out to be a fault in others: the simplicity of iterative systems makes it difficult or even impossible to speak of those properties of computing systems which are strongly related to their internal structure. However, this difficulty may be overcome by admitting additional structures within an iterative system, either on the set of states or on the transition function—or on both. A large part of the present paper treats of such a system, called a stored program computer.

1. Computations of iterative systems

Let $M = \langle X, \pi\rangle$ be an iterative system. An infinite sequence $(c_0, c_1, \ldots)$ of states of M will be called a *computation* of M iff $c_0 \in D(\pi)$ and for all $n \in N$, $c_{n+1} = \pi(c_n)$. Such a computation will be called *cyclic* when for some $i, j \in N$ such that $0 \leqslant i < j$ we have $c_i = c_j$. Obviously, we also have $c_{i+m} = c_{j+m}$ for all $m \in N$. A finite sequence $(c_0, \ldots, c_k)$ of states of M is a computation of M iff $c_0 \in D(\pi)$, $c_k \notin D(\pi)$ and for all $0 \leqslant i < k$, $c_{i+1} = \pi(c_i)$.

We shall call the first element of any computation (or sequence, in general) its *initial state*, and when the sequence is finite its last element will be called the *final state*. It may be observed that no state can be both initial and final. Thus any computation is an at least two-element sequence.

We shall often denote a computation with an initial state c_0 by $\bar{c}_0$, i.e.

$$\bar{c}_0 = (c_0, \pi(c_0), \pi^2(c_0), \ldots).$$

If $c = (c_0, c_1, \ldots)$ is any finite or infinite sequence, then for any $i \in N$ (when the sequence is infinite) or for $i = 0, 1, \ldots, l(c)-2$ (when it is finite) we shall

denote by $c^{(i)}$ the sequence $(c_i, c_{i+1}, \ldots)$. If c is a computation, then this sequence is again a computation with the initial state c_i and we shall call it a *segment* of c.

Let C_M be the set of all computations of the iterative system M. Observe that if we know the set C_M, it is possible to deduce all the information on M which concerns the active states of M, i.e. states in the set $D(\pi) \cup R(\pi)$. Clearly, the remaining states—let us call them *isolated states*—are of little interest when we use iterative systems to describe computing devices, thus we may often use the set C_M instead of M to speak of some properties of the system.

Let D be a set of at least two-element (maybe infinite) sequences. We shall call D a *set of computations* iff $D = C_M$ for some iterative system M. The following theorem fully characterizes sets of computations.

THEOREM 1.1. *Let $D \subset X^\infty$ be a set of at least two-element sequences. Then there exists an iterative system M such that $D = C_M$ if and only if the following conditions are satisfied*:

(i) *if $(c_0, c_1, \ldots)$ and $(d_0, d_1, \ldots)$ are in D and $c_i = d_j$ [1] for some $i, j \in N$, then $c_{i+1} = d_{j+1}$,*

(ii) *for all $c \in D$ and all $0 \leqslant i < l(c)-2$, $c^{(i)} \in D$.*

Proof. When D is a set of computations of some iterative system $M = \langle X, \pi\rangle$, then conditions (i) and (ii) easily follow from the fact that π is a (partial) function and from the definition of a segment of a computation.

Suppose D is a set of at least two-element sequences over X and let conditions (i) and (ii) be satisfied. We shall define an iterative system $M = \langle X, \pi\rangle$ as follows: let

$$D(\pi) = \{x \in X\colon x \text{ is an initial state of some } d_x \in D\}.$$

Observe that by (i) the assignment $x \mapsto d_x$ is a one-to-one correspondence. Let π be defined as follows: for any $x \in D(\pi)$, $\pi(x)$ is the successor of x in the sequence d_x (which by assumption has at least two elements). It follows from (i) that π is a well-defined function in X. We shall prove that $D = C_M$.

From the definition of M and using (ii) we infer that $D \subset C_M$. Suppose $\bar{c}_0$ is a computation in M. Since $c_0 \in D(\pi)$, it is an initial state of some sequence $d_{c_0} \in D$. It easily follows from (i), (ii) and the definition of π that $\bar{c}_0 = d_{c_0}$, thus $\bar{c}_0 \in D$. Hence $D = C_M$. ∎

COROLLARY 1.1. *Any subset of a set of computations satisfies* (i). ∎

COROLLARY 1.2. *Any subset of a set of computations is itself a set of computations if and only if it satisfies* (ii). ∎

It also follows from Theorem 1.1 that any set of computations defines a unique—up to isolated states—iterative system which generates it.

[1] Here (and in the sequel) equality means that either both sides are defined and equal or they are both undefined.

THEOREM 1.2. *Let D be any set of computations. If $D = C_{M_0}$ and $D = C_{M_1}$, where $M_0 = \langle X_0, \pi_0\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$, then $\pi_0 = \pi_1$, i.e. $D(\pi_0) = D(\pi_1)$, $R(\pi_0) = R(\pi_1)$ and for all $x \in D(\pi_0)$, $\pi_0(x) = \pi_1(x)$.*

Proof. Suppose that for some systems M_0 and M_1 we have $D = C_{M_0} = C_{M_1}$ and $\pi_0 \neq \pi_1$, i.e. for some $x \in D(\pi_0) \cup D(\pi_1)$, $\pi_0(x) \neq \pi_1(x)$. If $x \in D(\pi_i) - D(\pi_{1-i})$ $(i = 0, 1)$, then $\bar{x} \in C_{M_i}$, while no computation in M_{1-i} starts with x, hence $C_{M_i} \neq C_{M_{1-i}}$ contrary to our assumptions. On the other hand, if x belongs both to $D(\pi_0)$ and to $D(\pi_1)$, then $(x, \pi_0(x), \ldots) \in C_{M_0}$ and $(x, \pi_1(x), \ldots) \in C_{M_1}$. Assuming $C_{M_0} = C_{M_1}$, we obtain a contradiction with condition (i) of Theorem 1.1. Hence follows $\pi_0 = \pi_1$. ■

It is a simple consequence of Theorem 1.2 that the synthesis problem for iterative systems, i.e. finding for any set of computations a system which generates it, has a least solution: for any set of computations D there exists an iterative system $M = \langle X, \pi\rangle$ such that $D = C_M$, and for any other system $M_1 = \langle X_1, \pi_1\rangle$ such that $D = C_{M_1}$ the following holds: $\pi = \pi_1$ and $X \subset X_1$. This least system is $\langle D(\pi) \cup R(\pi), \pi\rangle$ where π is the partial function defined (uniquely, by Theorem 1.2) in the proof of Theorem 1.1.

We shall now consider the effect of Boolean operations performed on sets of computations.

THEOREM 1.3. *For any sets of computations D_1 and D_2, $D_1 \cap D_2$ is a set of computations.*

Proof. Let $d \in D_1 \cap D_2$. Since d is both in D_1 and in D_2, so are all its segments. Thus condition (ii) of Theorem 1.1 is fulfilled, which by Corollary 1.2 (since $D_1 \cap D_2 \subset D_1$) suffices to prove the theorem. ■

The sum of two sets of computations is not necessarily itself such a set. The following simple example illustrates the point.

Let $x_1 \neq x_2$; then the sets $\{(x, x_1)\}$ and $\{(x, x_2)\}$ are two sets of computations, each with one two-element computation. Clearly, the set $\{(x, x_1), (x, x_2)\}$ does not satisfy condition (i) of Theorem 1.1 and thus is not a set of computations. However, in some cases the sum of sets of computations is again a set of computations.

Let $M_0 = \langle X_0, \pi_0\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be arbitrary iterative systems. We shall call the systems M_0 and M_1 *consistent* iff $D(\pi_{1-i}) \cap (R(\pi_i) - D(\pi_i)) = \emptyset$ for $i = 0, 1$ and for all $x \in D(\pi_0) \cap D(\pi_1)$, $\pi_0(x) = \pi_1(x)$. Thus two systems are consistent if their transition functions coincide in their common domain and neither of the systems extends computations of the other.

LEMMA 1.1. *Systems $M_0 = \langle X_0, \pi_0\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ are consistent if and only if for all $x \in (D(\pi_0) \cup R(\pi_0)) \cap (D(\pi_1) \cup R(\pi_1))$, $\pi_0(x) = \pi_1(x)$.*

Proof. Let $A = (D(\pi_0) \cup R(\pi_0)) \cap (D(\pi_1) \cup R(\pi_1))$. Suppose M_0 and M_1 are consistent and let $x \in A$. It is easy to prove by the definition of consistency that $x \in D(\pi_0)$ if and only if $x \in D(\pi_1)$. Thus either $x \in D(\pi_0) \cap D(\pi_1)$ or $x \notin D(\pi_0) \cup D(\pi_1)$. In the former case, both $\pi_0(x)$ and $\pi_1(x)$ are defined and equal (by the consistency of M_0 and M_1), while in the latter they are both undefined. Hence, for all $x \in A$, $\pi_0(x) = \pi_1(x)$.

On the other hand, assume that this condition holds. Clearly, if $x \in D(\pi_0) \cap D(\pi_1)$, then also $x \in A$ and $\pi_0(x) = \pi_1(x)$. Suppose now that $D(\pi_{1-i}) \cap (R(\pi_i) - D(\pi_i)) \neq \emptyset$ for $i = 0$ or $i = 1$, and let x be any element of this set. Thus $x \in A$; moreover, $\pi_{1-i}(x)$ is defined, while $\pi_i(x)$ is not; hence $\pi_{1-i}(x) \neq \pi_i(x)$, contrary to our assumptions. Thus M_0 and M_1 must be consistent. ■

THEOREM 1.4. *Let D_0 and D_1 be sets of computations and let $M_0 = \langle X_0, \pi_0\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be iterative systems generating D_0 and D_1, respectively. Then $D_0 \cup D_1$ is a set of computations if and only if M_0 and M_1 are consistent.*

Proof. Assume that $D_0 \cup D_1$ is a set of computations. Again let $A = (D(\pi_0) \cup R(\pi_0)) \cap (D(\pi_1) \cup R(\pi_1))$ and suppose $x \in A$. Thus there exist computations $d^0 = (d_0^0, d_1^0, \ldots)$ in M_0 and $d^1 = (d_0^1, d_1^1, \ldots)$ in M_1 such that $x = d_i^0 = d_j^1$ for some $i, j \in N$. Since $d^0, d^1 \in D_0 \cup D_1$, we have, by Theorem 1.1, $\pi_0(x) = d_{i+1}^0 = d_{j+1}^1 = \pi_1(x)$, which proves by Lemma 1.2 that M_0 and M_1 are consistent.

Assume now that the systems M_0 and M_1 are consistent. Condition (ii) of Theorem 1.1 is obviously satisfied for $D_0 \cup D_1$, and so it remains to prove condition (i). Let $c = (c_0, c_1, \ldots)$ and $d = (d_0, d_1, \ldots)$ be two computations from the set $D_0 \cup D_1$ and suppose $c_i = d_j$ for some $i, j \in N$. If $c, d \in D_k$ for $k = 0$ or $k = 1$, then obviously $c_{i+1} = d_{j+1}$, D_k being a set of computations. If $c \in D_0$ and $d \in D_1$ (a similar argument holds for $c \in D_1$ and $d \in D_0$), then $c_i \in D(\pi_0) \cup R(\pi_0)$ and $d_j \in D(\pi_1) \cup R(\pi_1)$. Hence $c_i \in A$ and, M_0 and M_1 being consistent, $\pi_0(c_i) = \pi_1(d_j)$. Thus $c_{i+1} = d_{j+1}$. Hence $D_0 \cup D_1$ is a set of computations. ■

THEOREM 1.5. *For any two sets of computations D_0 and D_1, $D_0 - D_1$ is a set of computations if and only if for all $c, d \in D_0$, if $c \in D_0 \cap D_1$ and $c = d^{(1)}$, then $d \in D_0 \cap D_1$.*

Proof. Suppose $D_0 - D_1$ is a set of computations and let $c \in D_0 \cap D_1$, $c = d^{(1)}$ and $d \notin D_0 \cap D_1$. Hence $d \in D_0 - D_1$, while $d^{(1)}$ does not belong to $D_0 - D_1$, contrary to the assumption of $D_0 - D_1$ being a set of computations.

On the other hand, suppose that the condition of the theorem holds for all $c, d \in D_0$. Since $D_0 - D_1 \subset D_0$, we need to verify condition (ii) of Theorem 1.1. Suppose that for some $c \in D_0 - D_1$ and $i \geqslant 0$, $c^{(i)} \notin D_0 - D_1$. We may assume that i is the least integer for which this holds; clearly $i > 0$ and $c^{(i-1)}$ is defined. Let us denote this segment by d, i.e. $d = c^{(i-1)}$ and $d^{(1)} = c^{(i)}$. By assumption, $d \in D_0 - D_1$ and $d^{(1)} \in D_0 \cap D_1$. Thus we must have $d \in D_0 \cap D_1$, which gives a contradiction. ■

It is easy to see that if D is a set of computations and $D \subset XX^\infty$ for some non-empty set X, then the complement of D in XX^∞ is not a set of computations: if $x \in X$, then, for any set of computations D, $(x, x) \notin D$. Therefore $(x, x) \in XX^\infty - D$, which thereby cannot be a set of computations.

2. Simulation and homomorphisms

Let $M = \langle X, \pi \rangle$ be an iterative system. We shall define a partial function $f_M: X \to X$ as follows: for all $x, y \in X$, $f_M(x) = y$ iff there exists a finite computation $\bar{x}$ with a final state y. Thus, when $f_M(x)$ is defined, we may interpret it as the result of a computation with the initial state x. For this reason we shall call f_M the *result function* of the system M. Observe that $R(f_M)$ is the set of all final states of finite computations of M.

Let us state here an obvious property of this function:

LEMMA 2.1. $R(f_M) = R(\pi) - D(\pi)$. ■

Consider now the well-known problem of simulation: given two objects (usually able to perform some action), we are interested in interpreting one of them (or part of it) within the other and to deduce some properties of the object which is being simulated from the properties of the simulating object. If we assume that both objects can be described by iterative systems—and this assumption does not involve a great loss of generality for objects acting in discrete time—this problem turns into finding appropriate functions mapping one of the systems into the other.

In such a case, it seems that such a function should have the following property: if, starting a computation at some state x, we obtain a result y in one of the systems, then, interpreting the state x in the other system and starting a computation, we should obtain a result which corresponds to y. In other words, if $M = \langle X, \pi \rangle$ and $M_1 = \langle X_1, \pi_1 \rangle$ are two iterative systems, then a simulation function s from M into M_1 should have the following property:

$$s\big(f_M(x)\big) = f_{M_1}\big(s(x)\big) \tag{1}$$

for all $x \in D(f_M)$.

However, in most cases our interest lies also in the course of computations and not only in their results. Thus, in the most general case, we would expect with regard to s that the following hold: for every $x \in D(\pi)$ and $n \geqslant 1$ there exists some $m \geqslant 1$ such that

$$s\big(\pi^n(x)\big) = \pi_1^m\big(s(x)\big). \tag{2}$$

This condition ensures the preservation of succession of states by the function s.

Let us consider particular cases of (2). Suppose that for some $x \in D(\pi)$ such that $x \neq \pi(x)$ and $n \in N$ condition (2) holds with $n > m$. We may assume for simplicity that $n = 2$ and $m = 1$, i.e.

$$s\big(\pi^2(x)\big) = \pi_1\big(s(x)\big).$$

This means that (the succession of states being preserved) either $s\big(\pi(x)\big) = s(x)$ or $s\big(\pi(x)\big) = s\big(\pi^2(x)\big)$, i.e. the state $\pi(x)$ is undistinguishable in M_1 from one of the states $s(x)$, $s\big(\pi^2(x)\big)$. Hence we can omit $\pi(x)$ in M and still have the image of M in M_1 unchanged, the state $\pi(x)$ being irrelevant for simulation. In conclusion, we may assume that for all $x \in D(\pi)$, condition (2) holds with $n \leqslant m$. This in turn

is easily reduced to the following condition: for every $x \in D(\pi)$ there exists an $m \geqslant 1$ such that

$$s\big(\pi(x)\big) = \pi_1^m\big(s(x)\big). \tag{3}$$

The above considerations lead to the following definitions: let $M = \langle X, \pi \rangle$ and $M_1 = \langle X_1, \pi_1 \rangle$ be iterative systems and let $s: X \to X_1$ be a mapping. We shall call s a *simulation* iff conditions (1) and (3) hold for s. A simulation s will be called a *homomorphism* iff condition (3) holds with $m = 1$ for all $x \in D(\pi)$. Observe here that condition (1) may be replaced in the definition of simulation by the following one:

$$s\big(R(f_M)\big) \subset R(f_{M_1}).$$

Clearly, any theory of simulations includes that of homomorphisms, which are a particular case of the former. However, we shall proceed to demonstrate that any simulation may be replaced by a suitable homomorphism.

Suppose s is a simulation of a system $M = \langle X, \pi \rangle$ into a system $M_1 = \langle X_1, \pi_1 \rangle$. We shall extend the system M to a system $\overline{M} = \langle \overline{X}, \overline{\pi} \rangle$ as follows:

Let $x \in D(\pi)$ and suppose $s\big(\pi(x)\big) = \pi_1^m\big(s(x)\big)$ for $m > 0$. We add $m-1$ new states $x^1, x^2, \ldots, x^{m-1}$ (not belonging to X and different for all $x \in D(\pi)$) and we define the function $\overline{\pi}$ on them so that $\overline{\pi}^i(x) = x^i$ for $i = 1, \ldots, m-1$ and $\overline{\pi}^m(x) = \pi(x)$. Let $\overline{X}$ be the set X together with all states added for any $x \in D(\pi)$. We also assume that for any $x \in X$, $x \in D(\overline{\pi})$ iff $x \in D(\pi)$; thus $D(\overline{\pi}) = D(\pi) \cup (\overline{X} - X)$. It is easily verified that $\overline{\pi}$ is well defined in $\overline{X}$. Moreover, for any $x \in X$, $f_M(x) = f_{\overline{M}}(x)$ (see footnote on page 105). Thus, $D(f_M) \subset D(f_{\overline{M}})$ and $R(f_M) \subset R(f_{\overline{M}})$. Besides, if $x \in R(f_{\overline{M}})$, then by the construction of $\overline{M}$ there exists a state $x_1 \in X$ such that $f_{\overline{M}}(x_1) = x = f_M(x_1)$; hence,

$$R(f_M) = R(f_{\overline{M}}). \tag{4}$$

Next, let $\bar{s}$ be an extension of s over the set $\overline{X}$ defined as below:

for all $x \in X$, $\bar{s}(x) = s(x)$;

if $a \in \overline{X} - X$ and $a = x^i$ for some (exactly one) $x \in D(\pi)$ and $i > 0$, then let $\bar{s}(a) = \pi_1^i\big(s(x)\big)$.

$\bar{s}$ is a homomorphism of $\overline{M}$ into M, because: from (4) we obtain that

$$\bar{s}\big(R(f_{\overline{M}})\big) = s\big(R(f_M)\big) \subset R(f_{M_1})$$

since $\bar{s}|X = s$ and s is a simulation; also

$$\bar{s}\big(\overline{\pi}(x)\big) = \pi_1\big(\bar{s}(x)\big)$$

easily follows for all $x \in D(\overline{\pi})$ from the construction of $\overline{\pi}$ and $\bar{s}$. ■

The above construction shows that any simulation may be replaced by a suitable homomorphism, i.e. homomorphisms can "simulate" simulations. Moreover, the system $\overline{M}$ carries some additional information on how the original simulation has been performed.

3. Some properties of homomorphisms

Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be two iterative systems and suppose h is a homomorphism of M into M_1. It easily follows from the definition of homomorphisms that h preserves the character of computations in M, i.e.

THEOREM 3.1. *If $\bar{c}_0 = (c_0, c_1, \ldots)$ is a (finite, infinite, cyclic) computation, then the sequence $\overline{h(\bar{c}_0)} = (h(c_0), h(c_1), \ldots)$ is a (respectively: finite, infinite, cyclic) computation in M_1.*

Proof. From (3) (with $m = 1$ for all $x \in D(\pi)$) we infer that $c_0 \in D(\pi)$ implies $h(c_0) \in D(\pi_1)$. Moreover, for all $0 \leqslant i \leqslant l(\bar{c}_0)-2$, since $\pi(c_i) = c_{i+1}$, we obtain $\pi_1(h(c_i)) = h(\pi(c_i)) = h(c_{i+1})$.

If $c_n \notin D(\pi)$ for some $n \geqslant 1$, then $c_n \in R(f_M)$ and consequently $h(c_n) \in R(f_{M_1})$, i.e. it is a final state in the sequence $\overline{h(\bar{c}_0)}$. Clearly, if $c_i = c_j$ for some $0 \leqslant i < j$, then also $h(c_i) = h(c_j)$. ■

A system $M_0 = \langle X_0, \pi_0\rangle$ will be called a *subsystem* of $M = \langle X, \pi\rangle$ iff $X_0 \subset X$ and $\pi_0 = \pi|X_0$. This implies in particular that the identity function id: $X_0 \to X$ is a homomorphism of M_0 into M. If $\langle X_0, \pi_0\rangle$ is a subsystem of $\langle X, \pi\rangle$, then we shall write $\langle X_0, \pi\rangle$ instead of $\langle X_0, \pi_0\rangle$, often identifying this subsystem with the set X_0. Thus X_0 is a subsystem of X iff $\pi(X_0) \subseteq X_0$.

Observe that if $M = \langle X, \pi\rangle$ is an iterative system, then every subset $A \subset X$ generates a subsystem $\langle X_0, \pi\rangle$ of M such that $X_0 = \{x \in X\colon (\exists a \in A)(\exists k \geqslant 0)(\pi^k(a) = x)\}$. A will be called a *generating set* for X_0; X_0 will be called the *closure* of A and will often be denoted by $\bar{A}$.

Let σ be a relation in the set of states X of the system $M = \langle X, \pi\rangle$ such that for any $x, y \in X$

$x \sigma y$ iff there exist $i, j \geqslant 0$ such that $\pi^i(x) = \pi^j(y)$, where both sides are defined.

This relation is an equivalence in X and its equivalence classes are subsystems of the system M. They will be called *maximal connected subsystems* (m.c.-subsystems, for short) and the system $M = \langle X, \pi\rangle$ will be called *connected* when $\sigma = X^2$.

We shall prove that any homomorphism of a system M into M_1 may be decomposed into homomorphisms between m.c.-subsystems of M and M_1 and vice versa.

LEMMA 3.1. *Let h be a homomorphism of a system $M = \langle X, \pi\rangle$ into $M_1 = \langle X_1, \pi_1\rangle$. For any* m.c.-*subsystem $S \subset X$ there exists an* m.c.-*subsystem $S_1 \subset X_1$ such that $h(S) \subset S_1$.*

Proof. If $\pi^i(x) = \pi^j(y)$ for some $x, y \in S$ and $i, j \geqslant 0$, then $\pi_1^i(h(x)) = h(\pi^i(x)) = h(\pi^j(y)) = \pi_1^j(h(y))$, which is sufficient to prove the lemma. ■

The next lemma is quite obvious:

LEMMA 3.2. *If h is a homomorphism of $M = \langle X, \pi\rangle$ into M_1 and X_0 is a subsystem of X, then $h|X_0$ is a homomorphism of $\langle X_0, \pi\rangle$ into M_1.*

Proof. The lemma follows from the fact that $h|X_0 = h \cdot \mathrm{id}$, where id: $X_0 \to X$ is the identity function, the composition of homomorphisms being again a homomorphism [22]. ■

LEMMA 3.3. *Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be arbitrary systems. There exists a homomorphism of M into M_1 if and only if for any* m.c.-*subsystem S in M there exists a homomorphism of $\langle S, \pi\rangle$ into M_1.*

Proof. If h is a homomorphism of M into M_1, then, by Lemma 3.2, $h|S$ is a homomorphism of $\langle S, \pi\rangle$ into M_1 for any m.c.-subsystem $S \subset X$. On the other hand, suppose h_S is a homomorphism of an m.c.-subsystem $\langle S, \pi\rangle$ into M_1 for any $S \subset X$. Then the mapping h such that $h = \bigcup h_S$ is a well-defined homomorphism of M into M_1. ■

From Lemmas 3.1 and 3.3 we immediately obtain the following

THEOREM 3.2. *Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be arbitrary iterative systems. There exists a homomorphism h of M into M_1 if and only if for any* m.c.-*subsystem S of M there is an* m.c.-*subsystem S_1 of M_1 such that there exists a homomorphism of $\langle S, \pi\rangle$ into $\langle S_1, \pi_1\rangle$.* ■

It is a classical result in universal algebra that given a homomorphism between two algebras, the image of a subalgebra of the first is a subalgebra of the second. This need not be true for iterative systems. An easy example is the case of an isolated state of the first system (this is a subsystem) being mapped onto a state which lies in the domain of the transition function of the second system. However, the following may be proved:

THEOREM 3.3. *If h is a homomorphism of a system $M = \langle X, \pi\rangle$ into $M_1 = \langle X_1, \pi_1\rangle$ such that for any $x \in X$, $x \in D(\pi)$ iff $h(x) \in D(\pi_1)$, then for any subsystem X' of X its image $h(X')$ is a subsystem of X_1.*

The easy proof is omitted. ■

A 1-1 homomorphism h of X onto X_1 will be called an *isomorphism of* $M = \langle X, \pi\rangle$ *onto* $M_1 = \langle X_1, \pi_1\rangle$ iff for all $x \in X$, $x \in D(\pi)$ iff $h(x) \in D(\pi_1)$. From Theorem 3.3 we obtain the following

COROLLARY 3.1. *Let h be an isomorphism of $M = \langle X, \pi\rangle$ onto $M_1 = \langle X_1, \pi_1\rangle$. If X' is a subsystem of M, then $h(X')$ is a subsystem of M_1.* ■

No additional assumptions on homomorphisms are needed to prove that counter-images of subsystems are subsystems again. We shall use a topological argument.

Let $M = \langle X, \pi\rangle$ be an arbitrary system. It may be easily verified that the family $\mathscr{S}_M$ of all subsystems of M is a topology in X, i.e. the empty set $\emptyset$ and X are subsystems, and this family is closed under arbitrary joins and arbitrary (not necessarily finite) meets.

THEOREM 3.4. *Any homomorphism of a system $M = \langle X, \pi\rangle$ into $M_1 = \langle X_1, \pi_1\rangle$ is a continuous mapping of the space $(X, \mathscr{S}_M)$ into $(X_1, \mathscr{S}_{M_1})$.*

Proof. Suppose $h: X \to X_1$ is a homomorphism and let *Int* be the interior operation in $(X_1, \mathscr{S}_{M_1})$. We shall prove that for any set $B \subset X_1$, $h^{-1}(Int(B))$ is an open set in $(X, \mathscr{S}_M)$.

Assume $x \in h^{-1}(Int(B))$ and $x \in D(\pi)$. Thus $h(x) \in Int(B)$ and $\pi_1(h(x)) \in Int(B)$, since $Int(B)$ is a subsystem of M_1 which is an open set. Hence $h(\pi(x)) \in Int(B)$ which in turn implies $\pi(x) \in h^{-1}(Int(B))$. In consequence, $h^{-1}(Int(B))$ is a subsystem of M, i.e., it is an open set. ■

COROLLARY 3.2. *Let $h: X \to X_1$ be a homomorphism of $M = \langle X, \pi\rangle$ into $M_1 = \langle X_1, \pi_1\rangle$. For any subsystem B of M_1, $h^{-1}(B)$ is a subsystem of M.* ■

4. Congruences

An iterative system being a partial algebra, various definitions of congruence may be used. Our definition will be that of a strong congruence (see e.g. [10]). Let $M = \langle X, \pi\rangle$ be an iterative system. An equivalence relation $\varrho \subset X^2$ is a *congruence* in M iff whenever $(x, y) \in \varrho$ and $x \in D(\pi)$, then also $y \in D(\pi)$ and $(\pi(x), \pi(y)) \in \varrho$. Thus if two elements are congruent, then either both lie in the domain of the transition function or for both this function is undefined.

Let K_M be the family of all congruences in a system $M = \langle X, \pi\rangle$. Set-theoretic inclusion orders this family and the identity congruence is the least element in the ordered set $(K_M, \subset)$.

LEMMA 4.1. *There exists a greatest element ϱ_0 in $(K_M, \subset)$.*

Proof. Observe that for any congruence ϱ in M and any states $x \in D(\pi)$ and $y \in X - D(\pi)$, $(x, y) \notin \varrho$. Thus, when $\emptyset \neq D(\pi) \neq X$, X^2 is not a congruence in M.

Let us introduce a function H_M (or simply H) on the set $D(f_M) \cup R(f_M)$ and ranging over the set N so that

$$H(x) = k \quad \text{iff} \quad \pi^k(x) \in R(f_M). \tag{1}$$

Define the relation ϱ_0 as follows:

$$(x, y) \in \varrho_0 \quad \text{iff} \quad [x, y \in D(f_M) \,\&\, H(x) = H(y)] \vee [x, y \in D(\pi) - D(f_M)] \vee [x, y \in X - D(\pi)].$$

This relation is an equivalence relation in M (observe that $X = D(f_M) \cup (D(\pi) - D(f_M)) \cup (X - D(\pi))$ and also a congruence: if $x, y \in D(f_M)$ and $H(x) = H(y)$ then obviously $H(\pi(x)) = H(\pi(y))$; also, if $x, y \in D(\pi) - D(f_M)$ then $\pi(x), \pi(y) \in D(\pi) - D(f_M)$.

Let ϱ be any congruence in M and assume $(x, y) \in \varrho$. If $x, y \notin D(\pi)$, then also $(x, y) \in \varrho_0$. Suppose now $x, y \in D(\pi)$: if x and y are in the set $D(\pi) - D(f_M)$, then again $(x, y) \in \varrho_0$; if $x, y \in D(f_M)$, then we must have $(\pi^{H(x)}(x), \pi^{H(x)}(y)) \in \varrho$ since ϱ is a congruence. Therefore, $\pi^{H(x)}(y) \notin D(\pi)$ and $H(y) \leqslant H(x)$. Similarly, $H(x) \leqslant H(y)$, which gives $H(x) = H(y)$ and consequently $(x, y) \in \varrho_0$. This proves that ϱ_0 is indeed the greatest congruence in M. ■

THEOREM 4.1. *$(K_M, \subset)$ is a complete lattice.*

Proof. Since there exists a greatest element in K_M, it suffices to prove that for any non-empty family $(\varrho_i)_{i \in I}$ of congruences in M there exists a greatest lower bound in K_M for that family. It is easily verified that the relation $\bigcap_{i \in I} \varrho_i$ is the required congruence in M. ■

Let $\cup$ and $\cap$ be the lattice operations of join and meet, respectively, in the lattice $(K_M, \subset)$. From the above theorem it follows that $(K_M, \cup, \cap)$ is a complete lattice for any system M. More can be said about this lattice if we consider the systems which have only finite computations and no isolated states.

THEOREM 4.2. *For any iterative system $M = \langle D(f_M) \cup R(f_M), \pi\rangle$ the lattice $(K_M, \cup, \cap)$ is distributive if and only if, for every $n \in N$, $\operatorname{card}(H^{-1}(n)) \leqslant 2$.*

Proof. Suppose that for some $n \in N$ there exist three different states $x_1, x_2, x_3 \in H^{-1}(n)$. Let $\varrho_1, \varrho_2, \varrho_3$ be the least congruences such that $(x_1, x_2) \in \varrho_1$, $(x_2, x_3) \in \varrho_2$, $(x_3, x_1) \in \varrho_3$. Such congruences exist, K_M being a complete lattice, $(x_1, x_3) \notin \varrho_2$ and $(x_3, x_1) \notin \varrho_1$. Then $(x_1, x_3) \notin (\varrho_1 \cap \varrho_3) \cup (\varrho_2 \cap \varrho_3)$, while by transitivity $(x_1, x_3) \in \varrho_1 \cup \varrho_2$ and thus $(x_1, x_3) \in (\varrho_1 \cup \varrho_2) \cap \varrho_3$. Hence

$$(\varrho_1 \cup \varrho_2) \cap \varrho_3 \neq (\varrho_1 \cap \varrho_3) \cup (\varrho_2 \cap \varrho_3).$$

On the other hand, assume $\operatorname{card}(H^{-1}(n)) \leqslant 2$ for every $n \in N$ and consider any three congruences $\varrho_1, \varrho_2, \varrho_3$. Suppose $(x, y) \in (\varrho_1 \cup \varrho_2) \cap \varrho_3$ and $x \neq y$. Since $x, y \in D(f_M) \cup R(f_M)$, we must have $H(x) = H(y)$. Moreover, x and y are the only states in the set $H^{-1}(H(x))$. This implies that either $(x, y) \in \varrho_1$ or $(x, y) \in \varrho_2$. In consequence, since $(x, y) \in \varrho_3$, also $(x, y) \in \varrho_1 \cap \varrho_3$ or $(x, y) \in \varrho_2 \cap \varrho_3$, i.e., $(x, y) \in (\varrho_1 \cap \varrho_3) \cup (\varrho_2 \cap \varrho_3)$. The inverse inclusion is quite obvious. Thus,

$$(\varrho_1 \cup \varrho_2) \cap \varrho_3 = (\varrho_1 \cap \varrho_3) \cup (\varrho_2 \cap \varrho_3). \;\blacksquare$$

THEOREM 4.3. *Let $M = \langle D(f_M) \cup R(f_M), \pi\rangle$ and suppose that $(K_M, \cup, \cap)$ is a distributive lattice. Then $(K_M, \cup, \cap)$ is a Boolean algebra if and only if for any $n \in N$, $\operatorname{card}(H^{-1}(n) \cap R(\pi)) \leqslant 1$.*

Proof. Suppose there exists an $n \in N$ such that $\operatorname{card}(H^{-1}(n) \cap R(\pi))$ is equal to 2 (by Theorem 4.2, it cannot be greater than 2). Suppose that $x_3, x_4 \in H^{-1}(n) \cap R(\pi)$ $(x_3 \neq x_4)$ and let $x_1 \in \pi^{-1}(x_3)$ and $x_2 \in \pi^{-1}(x_4)$. Let ϱ be the least congruence such that $(x_3, x_4) \in \varrho$. Hence $(x_1, x_2) \notin \varrho$. Suppose ϱ_1 is a congruence in M such that $\varrho \cup \varrho_1$ is the greatest congruence in K_M. Thus $(x_1, x_2) \in \varrho_1$ and, ϱ_1 being a congruence, $(x_3, x_4) \in \varrho_1$. This implies that $\varrho \subset \varrho_1$ and $\varrho \cap \varrho_1$ is not the identity congruence in M. Therefore ϱ has no complement in K_M, which thereby is not a Boolean algebra.

To prove the sufficiency of the condition assume that ϱ is an arbitrary congruence in M. Let $\{E_i\}_{i \in I}$ be all the two-element equivalence classes of ϱ and let $N_0 = N - \bigcup_{i \in I} H(E_i)$. We shall define a relation $\bar{\varrho}$ in M so that

$$(x, y) \in \bar{\varrho} \quad \text{iff} \quad x = y \vee (H(x) = H(y) \wedge H(x) \in N_0).$$

$\bar{\varrho}$ is an equivalence relation. Suppose $(x, y) \in \bar{\varrho}$. Then either $x, y \notin D(\pi)$ (when $H(x) = 0$) or $x, y \in D(\pi)$ (when $H(x) > 0$). If the latter holds, then by assump-

tion we must have $\pi(x) = \pi(y)$, since $\pi(x)$, $\pi(y)$ are both in the set $H^{-1}\big(H(\pi(x))\big) \cap R(\pi)$. Thus $\big(\pi(x), \pi(y)\big) \in \bar{\varrho}$, which proves that $\bar{\varrho}$ is a congruence in M. It is easy to verify that $\bar{\varrho}$ is a complement for ϱ in the lattice $(K_M, \cup, \cap)$. This suffices to prove that $(K_M, \cup, \cap)$ is a Boolean algebra. ■

Let ϱ be a congruence in an iterative system $M = \langle X, \pi\rangle$. We can define a quotient system $M/\varrho = \langle X/\varrho, \pi^*\rangle$ as follows: X/ϱ is the set of all equivalence classes of ϱ and $D(\pi^*) = \{\alpha \in X/\varrho\colon \alpha \subset D(\pi)\}$ with $\pi^*([x]) = [\pi(x)]$. As usual, the natural mapping $\varepsilon_\varrho\colon X \to X/\varrho$ is a homomorphism of M into M/ϱ. Observe, however, that this statement remains true even for a differently defined transition function π^* in the quotient system, e.g., if $x \in X - \big(D(\pi) \cup R(\pi)\big)$, which means that x is an isolated state, then $[x]$ may belong to $D(\pi^*)$ and π^* may be defined anyhow on $[x]$. Thus the requirement that the natural mapping ε_ϱ be a homomorphism does not determine uniquely the transition function in the quotient system.

Let $M_i = \langle X_i, \pi_i\rangle$, $i \in I$, be a family of iterative systems. We shall define a system $M = \langle X, \pi\rangle$, called the *product* of the family $\{M_i\}_{i\in I}$, so that

$$X = \mathop{\mathsf{P}}_{i\in I} X_i, \quad D(\pi) = \mathop{\mathsf{P}}_{i\in I} D(\pi_i) \quad \text{and} \quad \pi\big((x_i)_{i\in I}\big) = \big(\pi_i(x_i)\big)_{i\in I}.$$

It should be remarked that projections $p_i\colon X \to X_i$ need not be homomorphisms. As a simple counterexample consider two systems, $M_1 = \langle\{a, b\}, \pi_1\rangle$ and $M_2 = \langle\{c\}, \pi_2\rangle$, such that $\pi_1(a) = b \notin D(\pi_1)$ and $\pi_2(c) = c$. Then (b, c) is a final state in the product system $M = \langle\{(a, c), (b, c)\}, \pi\rangle$ while $p_2(b, c) = c$ is not a final state in M_2. Therefore, given a family $M_i = \langle X_i, \pi_i\rangle$, $i \in I$, of systems, we shall say that a subsystem A of the product system $\mathop{\mathsf{P}}_{i\in I} M_i = \langle \mathop{\mathsf{P}}_{i\in I} X_i, \pi\rangle$ is a *subdirect product* iff all the projections p_i for $i \in I$ are onto mappings.

THEOREM 4.4. *Let $\{\varrho_i\}_{i\in I}$ be an arbitrary family of congruences in a system $M = \langle X, \pi\rangle$ and $\varrho = \bigcap_{i\in I} \varrho_i$. Then M/ϱ is isomorphic to a subdirect product of the systems M/ϱ_i, $i \in I$.*

Proof. Let $M/\varrho = \langle X/\varrho, \pi^*\rangle$, $M/\varrho_i = \langle X/\varrho_i, \pi_i\rangle$ and let $\mathop{\mathsf{P}}_{i\in I} M/\varrho_i = \langle \mathop{\mathsf{P}}_{i\in I} X/\varrho_i, \pi_*\rangle$ be the product of the systems M/ϱ_i, $i \in I$. Take $h([x]_\varrho) = ([x]_{\varrho_i})_{i\in I}$. It is easily verified that h is 1-1.

Observe that if $[x]_\varrho \notin D(\pi^*)$, then $x \notin D(\pi)$ and consequently $[x]_{\varrho_i} \notin D(\pi_i)$ for all $i \in I$, i.e. $([x]_{\varrho_i})_{i\in I} \notin D(\pi_*)$. Suppose $[x]_\varrho \in D(\pi^*)$. Then

$$h\big(\pi^*([x]_\varrho)\big) = h\big([\pi(x)]_\varrho\big) = \big([\pi(x)]_{\varrho_i}\big)_{i\in I} = \big(\pi_i([x]_{\varrho_i})\big)_{i\in I} = \pi_*\big(h([x]_\varrho)\big). \tag{2}$$

Moreover, if $[x]_\varrho \in R(f_{M/\varrho})$, then, by Lemma 2.1, $[x]_\varrho \in R(\pi^*) - D(\pi^*)$, which by the definition of a quotient system is equivalent to $x \in R(\pi) - D(\pi)$. Hence, for every $i \in I$, $[x]_{\varrho_i} \in R(\pi_i) - D(\pi_i)$, and thus $h([x]_\varrho) = ([x]_{\varrho_i})_{i\in I} \in R(\pi_*) - D(\pi_*)$, which proves that

$$h\big(R(f_{M/\varrho})\big) \subset R\big(f_{\mathop{\mathsf{P}}_{i\in I} M/\varrho_i}\big). \tag{3}$$

By (2) and (3), h is a homomorphism and, by Theorem 3.3, $h(X/\varrho)$ is a subsystem of $\mathop{\mathsf{P}}_{i\in I} M/\varrho_i$ isomorphic to X/ϱ. It is clear that, for all $i \in I$, $p_i\big(h(X/\varrho)\big) = X/\varrho_i$. ■

A congruence ϱ in M is said to be *meet-irreducible* iff for any family of congruences $\{\varrho_i\}_{i\in I}$, $\bigcap_{i\in I} \varrho_i = \varrho$ implies $\varrho = \varrho_i$ for some $i \in I$. A system M is *subdirectly irreducible* iff the identity congruence is meet-irreducible in M.

LEMMA 4.2. *Let ϱ be a congruence in a system $M = \langle X, \pi\rangle$. Then M/ϱ is subdirectly irreducible if and only if ϱ is meet-irreducible.*

Proof. Assume that $\varrho = \bigcap_{i\in I} \varrho_i$ and $\varrho \subsetneqq \varrho_i$ for all $i \in I$. Define a family of relations $\{\sigma_i\}_{i\in I}$ in M/ϱ as follows:

$$([x]_\varrho, [y]_\varrho) \in \sigma_i \quad \text{iff} \quad (x, y) \in \varrho_i.$$

Each σ_i is a non-identity congruence in M/ϱ. Moreover, $\bigcap_{i\in I} \sigma_i$ is the identity congruence, which implies that M/ϱ is not subdirectly irreducible.

Conversely, assume that M/ϱ is not subdirectly irreducible, i.e., for some family $\{\sigma_i\}_{i\in I}$ of non-identity congruences in M/ϱ the meet $\bigcap_{i\in I} \sigma_i$ is the identity congruence. Define ϱ_i, $i \in I$, so that

$$(x, y) \in \varrho_i \quad \text{iff} \quad ([x]_\varrho, [y]_\varrho) \in \sigma_i.$$

Thus $(x, y) \in \bigcap_{i\in I} \varrho_i$ iff $([x]_\varrho, [y]_\varrho) \in \bigcap_{i\in I} \sigma_i$, i.e. $(x, y) \in \bigcap_{i\in I} \varrho_i$ iff $(x, y) \in \varrho$. Therefore $\varrho = \bigcap_{i\in I} \varrho_i$, where $\varrho \subsetneqq \varrho_i$ (since σ_i are not identities), which implies that ϱ is not meet-irreducible. ■

Now we can prove Birkhoff's theorem for iterative systems.

THEOREM 4.5. *Every iterative system $M = \langle X, \pi\rangle$ is isomorphic to a subdirect product of subdirectly irreducible systems.*

Proof. Iterative systems being partial algebras, the classic proof of Birkhoff's theorem (see e.g. [7]) is not affected. Thus, let $\{\varrho_i\}_{i\in I}$ be the family of all meet irreducible congruences in M. Clearly, this family is non-empty, since it contains the greatest element in the lattice K_M. We claim that $\bigcap_{i\in I} \varrho_i$ is the identity congruence.

Take $x, y \in X$ and assume $x \neq y$. The join of any chain of congruences being again a congruence, we infer by the Kuratowski–Zorn lemma that there exists a maximal congruence ϱ_0 such that $(x, y) \notin \varrho_0$. Being maximal, ϱ_0 is clearly meet-irreducible, i.e. it belongs to the family $\{\varrho_i\}_{i\in I}$. Therefore $(x, y) \notin \bigcap_{i\in I} \varrho_i$ and $\bigcap_{i\in I} \varrho_i$ is the identity congruence. By Theorem 4.4, the system M (which is isomorphic to $M/\bigcap \varrho_i$) is isomorphic to a subdirect product of the systems M/ϱ_i which by Lemma 4.2 are subdirectly irreducible. ■

5. Another approach to simulation

We shall consider here a more general idea of simulation than that presented in Section 2. Defining a simulation, we meant a mapping which preserves the property of being a final state and also preserves the succession of states. However, it is often necessary in a simulation process to preserve some intrinsic properties of states, which cannot be described by the transition function of an iterative system. In this section we shall define and investigate some notions which will make it possible to speak of such cases also.

Let X, X_1 be arbitrary sets. We shall call any relation $v \subset X \times X_1$ a *property* and any family $V = \{v_i\}_{i \in I}$ of such relations a *set of properties* (in $X \times X_1$).

Let $M = \langle X, \pi \rangle$ and $M_1 = \langle X_1, \pi_1 \rangle$ be iterative systems. A *basis of simulation* of M in M_1 is any ordered pair $B = (v, V)$, where V is a set of properties in $X \times X_1$ and $v \in V$ (v is the *initial property* of the basis B). The properties in V will often be called simply *properties in the basis B*.

Let $B = (v, V)$ be a basis of simulation of M in M_1, where $V = \{v_i\}_{i \in I}$. We shall say that M is *B-simulated* by M_1 (and we shall write $M <_B M_1$) iff there exists a partial function $f \in X_1^{[X]}$ such that:

(i) $D(f) = \bigcup_{i \in I} D(v_i) \cap \overline{D(v)}$,

(ii) for any $i \in I$ and $x \in D(f)$, if $x \in D(v_i)$, then $\big(x, f(x)\big) \in v_i$,

(iii) for any computation $\bar{c}_0$ in M such that $c_0 \in D(v)$, $f(\bar{c}_0 | D(f))$ is a subsequence of some computation in M_1.

The relation $<_B$ will be called *B-simulation*. We shall sometimes write $M <_{B,f} M_1$ to stress that f is the function required by the definition of B-simulation. Thus f associates with every state in M endowed with a certain property some state in M_1 which has the same property. Moreover, if a computation starts in M with a state which has the initial property, then f, reduced to this computation, behaves partly as a simulation as defined in Section 2, i.e. within its domain it preserves the succession of states (though not necessarily the property of being a final state).

Let us consider two examples of such simulation.

EXAMPLE 1. *Functions computable by iterative systems.*

Let $M = \langle X, \pi \rangle$ and $M_1 = \langle X_1, \pi_1 \rangle$ be two iterative systems. Suppose I, O are finite sets (called *input* and *output alphabets*, respectively). Let $i\colon I^* \to X$ and $i_1\colon I^* \to X_1$ be 1-1 functions (called *coding functions*), while $o\colon X \to O^*$ and $o_1\colon X_1 \to O^*$ are functions called *decoding functions*.

We shall distinguish two properties in $X \times X_1$: v is a property of equal codes, i.e. $(x, x_1) \in v$ iff $i^{-1}(x) = i_1^{-1}(x_1)$ and v_0 is a similar property concerning the decoding functions reduced to the final states, i.e. $(x, x_1) \in v_0$ iff $(x, x_1) \in R(f_M) \times R(f_{M_1})$ and $o(x) = o_1(x_1)$.

Let us consider a basis of simulation defined as follows:

$$B = (v, \{v, v_0\}).$$

Then, if $M <_B M_1$ and $D(v) = R(i)$, we have for $x \in D(o \cdot f_M \cdot i)$,

$$(o \cdot f_M \cdot i)(x) = (o_1 \cdot f_{M_1} \cdot i_1)(x).$$

Thus with a given basis of simulation a function computable in M (for some fixed coding and decoding functions) is also computable in M_1. ■

EXAMPLE 2. *Simulation of memory contents.*

Let A and B be non-empty sets, called the *set of addresses* and the *alphabet*, respectively. Let us consider two iterative systems, $M = \langle X, \pi \rangle$ and $M_1 = \langle X_1, \pi_1 \rangle$, such that $X \subset B^A$ and $X_1 \subset B^A$. The elements of the sets X and X_1 are called *memory states* of M and M_1. For any $c \in X$ (or $c \in X_1$) the set $c(A)$ is *the set of memory contents* in the state c.

Suppose $\beta\colon A \to A$ is a 1-1 function. For any $c \in X$ let c_β be a function from $R(\beta)$ into B such that for any $a \in R(\beta)$,

$$c_\beta(a) = c\big(\beta^{-1}(a)\big).$$

We define a property $v_\beta \subset X \times X_1$ as follows: for any $c \in X$ and $c_1 \in X_1$,

$$(c, c_1) \in v_\beta \quad \text{iff} \quad c_1 | R(\beta) = c_\beta.$$

Thus when $M <_{B,f} M_1$ with $B = (v, V)$, $v_\beta \in V$ and $D(v) = X$, the function f associates with $c \in X$ a certain state $f(c) \in X_1$ so that all the memory contents of c are preserved in $f(c)$ and $c(a) = f(c)\big(\beta(a)\big)$ for any $a \in A$. ■

We have proved in Section 1 that for any set of computations D there exists a least iterative system M such that $D = C_M$ (Theorem 1.2). Moreover, if D and D_1 are two sets of computations and $D = C_M$ and $D_1 = C_{M_1}$ (M and M_1 being the least such systems), then there exists a (least) system with a set of computations $D \cap D_1$ (Theorem 1.3). This system will be denoted by $M \wedge M_1$. Observe that $M \wedge M_1$ may be considered to be a subsystem of both M and M_1.

Similarly, if M and M_1 are consistent, then there exists a least system with a set of computations $D \cup D_1$ (Theorem 1.4): we shall denote this system by $M \vee M_1$.

The following lemma is an easy consequence of the definition of B-simulation.

LEMMA 5.1. *Let B be a basis of simulation of a system M in another system M_1 such that $M <_B M_1$. Then for any subsystem M' of M, $M' <_B M_1$.* ■

Using this lemma we immediately obtain

THEOREM 5.1. *If $M <_B M_1$, then, for any iterative system M_0, $M \wedge M_0 <_B M_1$.*

THEOREM 5.2. *If $M <_B M'$ and $M_1 <_B M_1'$ and for any computation $\bar{x}_0$ in M' such that $x_0 \in R(v)$, where v is the initial property in B, we also have $\bar{x}_0 \in C_{M_1'}$, then*

$$M \wedge M_1 <_B M' \wedge M_1'.$$

Proof. From the previous theorem we have $M \wedge M_1 <_{B,f} M'$ for some partial function f. It is now easy to check that under the assumptions of the theorem we also have $M \wedge M_1 <_{B,f} M' \wedge M_1'$. ■

THEOREM 5.3. *If $M <_B M'$ and $M_1 <_B M_1'$, M and M_1 as well as M' and M_1' are consistent systems, then*

$$M \vee M_1 <_B M' \vee M_1'.$$

Proof. If $M <_{B,f} M'$ and $M_1 <_{B,f_1} M_1'$, then $M \vee M_1 <_{B,f\cup f_1} M' \vee M_1'$. ■

We shall now consider conditions which make the relation of simulation transitive.

If $B = (v, V)$ and $B_1 = (v_1, V_1)$ are simulation bases, then $B \cdot B_1$ denotes their composition, i.e. a simulation basis of the form $B \cdot B_1 = (v \cdot v_1, V \cdot V_1)$, where $V \cdot V_1 = \{v' \cdot v'' \colon v' \in V \text{ and } v'' \in V_1\}$.

THEOREM 5.4. *Let M, M_1 and M_2 be iterative systems and assume that $B = (v, V)$ and $B_1 = (v_1, V_1)$ are bases of simulation of M in M_1 and of M_1 in M_2, respectively, such that, for any $v' \in V$ and $v'' \in V_1$, if $R(v') \cap D(v'') \neq \emptyset$, then $R(v') \subset D(v'')$. Then $M <_B M_1$ and $M_1 <_{B_1} M_2$ imply $M <_{B\cdot B_1} M_2$.*

Proof. The theorem holds when $R(v) \cap D(v_1) = \emptyset$, since $v \cdot v_1 = \emptyset$ and therefore $\overline{D(v \cdot v_1)} = \emptyset$.

Suppose thus that $R(v) \cap D(v_1) \neq \emptyset$ and let f and f_1 be partial functions such that

$$M <_{B,f} M_1 \tag{1}$$

and

$$M_1 <_{B_1,f_1} M_2. \tag{2}$$

Let $Y = \bigcup_{\substack{v\in V' \\ v_1'\in V_1}} D(v' \cdot v_1') \cap \overline{D(v \cdot v_1)}$, where the closure of $D(v \cdot v_1)$ is taken in M. We shall prove that $M <_{B\cdot B_1, f_2} M_2$, where $f_2 = f_1 \cdot f | Y$. Observe that $D(f_2) = Y$, since for any $v' \in V$ and all $v'' \in V_1$, $D(v' \cdot v'') \subset D(v')$ we have $Y \subset D(f)$. Moreover, $R(f|Y) \subset D(f_1)$.

Consider an arbitrary computation $\bar{x}_0 = (x_0, x_1, \ldots)$ in M with $x_0 \in D(v \cdot v_1)$. Let x_i be a state of the computation $\bar{x}_0$. If $x_i \in D(v'')$ for $v'' \in B \cdot B_1$, then for some $v_1'' \in B$ and $v_2'' \in B_1$ we have

$$v'' = v_1'' \cdot v_2''. \tag{3}$$

Thus $x_i \in D(v_1'')$, and by (1)

$$\big(x_i, f(x_i)\big) \in v_1''. \tag{4}$$

Since $x_i \in D(v'')$, we have $R(v_1'') \cap D(v_2'') \neq \emptyset$. Thus, using the assumptions of the theorem, we obtain $f(x_i) \in D(v_2'')$. Hence we infer from (2) that

$$\big(f(x_i), f_1(f(x_i))\big) \in v_2''. \tag{5}$$

Thus, by (3), (4) and (5), $\big(x_i, (f_1 \cdot f)(x_i)\big) \in v''$, which proves that condition (ii) of the definition of simulation is satisfied.

It is also easy to verify that, by (1) and (2) and condition (iii) in the definition of simulation, $f_2(\bar{x}_0|Y)$ is a subsequence of some computation in M_2. In consequence, $M <_{B\cdot B_1, f_2} M_2$. ■

THEOREM 5.5. *Let M, M_1 and M_2 be iterative systems with a common set of states X and suppose $B = (v, V)$ is a basis of simulation such that all properties in V are equivalences in X. Then $M <_B M_1$ and $M_1 <_B M_2$ imply $M <_B M_2$.*

Proof. Suppose $M <_{B,f} M_1$ and $M_1 <_{B,f_1} M_2$. Let $Y = \bigcup_{v\in V} D(v) \cap \overline{D(v)}$, where the closure of $D(v)$ is in M. We shall prove that $M <_{B,f_2} M_2$, where $f_2 = f_1 \cdot f | Y$. Observe that again $D(f_2) = Y$.

Assume that $\bar{x}_0 = (x_0, x_1, \ldots)$ is a computation in M with $x_0 \in D(v)$. If x_i is an arbitrary state of the computation $\bar{x}_0$ and $x_i \in D(v')$ for some $v' \in V$, then $\big(x_i, f(x_i)\big) \in v'$; v' being an equivalence; also $\big(f(x_i), x_i\big) \in v'$. Thus $f(x_i) \in D(v')$ and $\big(f(x_i), f_1(f(x_i))\big) \in v'$. By the transitivity of v' we have then $\big(x_i, (f_1 \cdot f)(x_i)\big) \in v'$. It is easy to observe that the sequence $f_2(\bar{x}_0|Y)$ is a subsequence of some computation in M_2. Thus $M <_{B,f_2} M_2$. ■

Given a basis of simulation B of one iterative system in another, it may sometimes be useful to find another basis B_1, in some sense simpler than the given one, and such that properties in B may be obtained from those in B_1.

THEOREM 5.6. *Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be iterative systems and suppose that $B = (v, V)$, where $V = \{v_i\}_{i\in I}$, is a basis of simulation of M in M_1 such that for any property $v_i \in V$ and arbitrary $x \in X$ and $x_1 \in X_1$, if $(x, x_1) \in v_i$, then*

$$\{i \in I \colon x \in D(v_i)\} = \{i \in I \colon x_1 \in R(v_i)\} \tag{6}$$

and for all $v_i \neq v$, $i \in I$, $D(v_i) \cap D(v) = \emptyset$. Then there exists a basis of simulation $B' = (v', V')$ such that:

(i) *if $M <_B M_1$, then $M <_{B'} M_1$;*

(ii) *the properties in B' are disjoint, i.e. for any v_1, v_2 in V', if $v_1 \neq v_2$, then $D(v_1) \cap D(v_2) = \emptyset$;*

(iii) *for every property $v_i \in V$ there exists a subset $V_i' \subset V'$ such that $v_i = \bigcup_{v''\in V_i'} v''$.*

Proof. For any $T \subset I$ let v_T be a property in $X \times X_1$ such that for all $x \in X$ and $x_1 \in X_1$

$$(x, x_1) \in v_T \quad \text{iff} \quad \big((x, x_1) \in v_i \text{ if and only if } i \in T\big).$$

Let V' be the set of all non-empty properties v_T for all $T \subset I$, i.e.

$$V' = \{v_T \colon T \subset I\} - \{\emptyset\}.$$

Properties in V' are disjoint. Indeed, suppose $T \neq T'$ and $x \in D(v_T) \cap D(v_{T'})$. There exist some $x_1, x_2 \in X_1$ such that $(x, x_1) \in v_T$ and $(x, x_2) \in v_{T'}$. T' being non-empty, there exists some property v_i with $i \in T'$ such that $(x, x_2) \in v_i$. Moreover, if $i_0 \in T - T'$ (analogously for $i_0 \in T' - T$) then $(x, x_1) \in v_{i_0}$ and $(x, x_2) \notin v_{i_0}$.

Hence $i_0 \in \{i \in I: x \in D(v_i)\}$ and $i_0 \notin \{i \in I: x_2 \in R(v_i)\}$, which contradicts (6). This proves (ii).

By the definition of V', for every property $v_i \in V$, $v_i = \bigcup_{D(v_T) \subset D(v_i)} v_T$. Assume $(x, x_1) \in v_i$ and let $T = \{j \in I: (x, x_1) \in v_j\}$; then clearly $(x, x_1) \in v_T$ and $D(v_T) \subset D(v_i)$. Hence $v_i \subset \bigcup_{D(v_T) \subset D(v_i)} v_T$. The second inclusion is obvious.

Since $D(v_i) \cap D(v) = \emptyset$ for all $v_i \neq v$, the property v belongs to V': if $v = v_i{}^0$ for some $i_0 \in I$, then $v = v_{|i_0|}$. Let the basis B' be defined as follows:

$$B' = (v, V').$$

We shall prove that B' is a basis satisfying (i) (by the above argument it satisfies (ii) and (iii)). Suppose $M <_{B,f} M_1$ and let $\overline{x}_0 = (x_0, x_1, \ldots)$ be a computation in M with $x_0 \in D(v)$. Consider an arbitrary state x_i of this computation such that $x_i \in D(v_T)$ for some $T \subset I$. Since for all $t \in T$, $x_i \in D(v_t)$, we have $(x_i, f(x_i)) \in v_t$ for $t \in T$. By (6), also $(x_i, f(x_i)) \in v_T$.

We also infer that $f(\overline{x}_0|Y)$ is a subsequence of some computation in M_1, where $Y = \bigcup_{v' \in V'} D(v') \cap \overline{D(v)}$ and the closure of $D(v)$ is taken in M. This completes the proof of the fact that $M <_{B',f} M_1$. ■

Given two iterative systems M and M_1 such that $M <_B M_1$ for some simulation basis B, it may be important to find a simulation basis induced by B which would ensure the simulation between quotient systems obtained by some congruences in M and M_1. This may be the case when we look for simulations preserving some congruences. We shall proceed to construct such a basis.

Let $M = \langle X, \pi \rangle$ and $M_1 = \langle X_1, \pi_1 \rangle$ be iterative systems and ϱ, ϱ_1 congruences in M and M_1, respectively. Suppose $B = (v, V)$ is a simulation basis of M in M_1. For any property $v' \in V$ we define a property $\overline{v}'$ in $X/\varrho \times X_1/\varrho_1$, called an *adjoint property with respect to* v', such that, for any $\alpha \in X/\varrho$ and $\beta \in X_1/\varrho_1$, $(\alpha, \beta) \in \overline{v}'$ iff there exist $x \in \alpha$ and $y \in \beta$ such that $(x, y) \in v'$.

A basis of simulation $\overline{B} = (\overline{v}, \overline{V})$, where $\overline{v}$ is the property adjoint to v and $\overline{V}$ is the set of all properties adjoint to properties in V, will be called the *simulation basis adjoint to* B.

THEOREM 5.7. *Let ϱ and ϱ_1 be congruences in iterative systems $M = \langle X, \pi \rangle$ and $M_1 = \langle X_1, \pi_1 \rangle$, respectively, and suppose B is a simulation basis of M in M_1. Assume that for any property v in B and, for any $x, y \in X$ and $x_1, y_1 \in X_1$,*

(i) *if $(x, y) \in \varrho$, $(x_1, y_1) \in \varrho_1$ and $(x, x_1) \in v$, then $(y, y_1) \in v$,*

(ii) *if $(x, y) \in \varrho$, $(x, x_1) \in v$ and $(y, y_1) \in v$, then $(x_1, y_1) \in \varrho_1$.*

Then $M <_B M_1$ implies $M/\varrho <_{\overline{B}} M_1/\varrho_1$.

Proof. Suppose $M <_{B,f} M_1$. Let $\overline{f}$ be a partial function from X/ϱ into X_1/ϱ_1 such that

(a) if $\alpha \in X/\varrho$, then $\alpha \in D(\overline{f})$ iff there exists an $x \in \alpha$ such that $x \in D(f)$;

(b) if $\alpha \in X/\varrho$, then $\overline{f}(\alpha) = [f(x)]_{\varrho_1}$ for an arbitrary element $x \in \alpha \cap D(f)$.

$\overline{f}$ is a well-defined function. Indeed, suppose $x, y \in \alpha \cap D(f)$. Then $(x, y) \in \varrho$ and $x \in D(v)$ for some property v in B, since $x \in D(f)$. Hence, by the definition of B-simulation, $(x, f(x)) \in v$. We now have $(x, y) \in \varrho$, $(f(x), f(x)) \in \varrho_1$ and $(x, f(x)) \in v$. Therefore, by (i), also $(y, f(x)) \in v$, which proves that $y \in D(v)$. Therefore $(y, f(y)) \in v$. Applying (ii), we infer that $(f(x), f(y)) \in \varrho_1$. This shows that $\overline{f}(\alpha)$ does not depend on the choice of elements from the set $\alpha \cap D(f)$. It may now be easily verified that $M/\varrho <_{\overline{B},\overline{f}} M_1/\varrho_1$. ■

II. STORED PROGRAM COMPUTERS

The considerations in this chapter concern properties of stored program computers (SPC) as defined in [21]. The model discussed here seems to be an adequate description of existing digital computers, being at the same time a relatively simple one as compared with other formal notions of a computer (see e.g. [1], [8], [9], [11], [33]). The notion of a program is introduced here as an object related to some computer. In fact, a program may be considered as a description of some part of potential computational capabilities of a computer. Hence investigations of program properties turn out to be investigations of computer properties as well.

In the present chapter only some properties of SPCs are presented. Other results (e.g. regarding properties of translators) may be found elsewhere (see bibliography in [16]).

In the first section the definitions of a SPC and a program are introduced, as well as some other basic notions related to them. In Section 2 it is proved that an instruction list of an SPC may in some cases be reduced without diminishing the computational power of the computer. Section 3 describes a method of comparing programs through iterative systems which they generate. In Section 4 decomposition of programs into subprograms is used for program simplification. In the last three sections a classification of programs is given. This classification is based on the length of computations of a given program, the memory occupied by these computations and the structure of programs.

1. Stored program computers and programs

In Chapter I we considered iterative systems as a very general model of digital computers. However, for many specific problems concerning computers this model is not rich enough. In this section we shall introduce iterative systems with additional structure imposed on the set of states and the transition function. Such systems seem to be a much closer description of existing computers of the von Neumann type and they will be called von Neumann computers or *stored program computers* (SPC). A stored program computer $M = \langle C, \pi \rangle$ is an iterative system specified as follows.

Let A be an arbitrary at most countable non-empty set (the set of addresses of M) and let $A \subset B$ (B is the alphabet of M). Then C is some subset of the set $B^{[A]}$ of all partial functions from A to B. C will be called the *memory* of M and its

elements will be called *memory states*. If $c \in C$ and $a \in D(c)$, then $c(a)$, if defined, is the contents of the address a in the memory state c.

In the set A we distinguish one address l, called the *instruction counter*, assuming $l \in D(c)$ and $c(l) \in A$ for all $c \in C$.

To define the transition function π of the system M, suppose there is a certain finite set of elementary operations $\{f_1, \ldots, f_n\}$ (i.e. f_i is a partial operation in B with arity $\mathrm{ar}(f_i)$) associated with M. Then we define an alphabet $\mathscr{A} = \{\mathfrak{f}_1, \ldots, \mathfrak{f}_n, \alpha, (,), ,\}$, where $\mathfrak{f}_1, \ldots, \mathfrak{f}_n$ are names for elementary operations in M. For every such name set $\mathrm{ar}(\mathfrak{f}_i) = \mathrm{ar}(f_i)$. The set of terms over $\mathscr{A}$ is the least set T such that

(i) $A \subset T$,

(ii) if $t \in T$, then $\alpha(t) \in T$,

(iii) if $\mathfrak{f}_i \in \mathscr{A}$ and $\mathrm{ar}(\mathfrak{f}_i) = m$, $t_1, \ldots, t_m \in T$, then $\mathfrak{f}_i(t_1, \ldots, t_m) \in T$.

The set $\mathscr{R} = \{t \to t' : t, t' \in T\}$ will be the *set of instructions* over T and some subset R of this set will be assigned to the stored program computer M. R will be called the *instruction list* of M.

Terms and instructions being defined as syntactic entities, we define a partial valuation function $v_c : T \to B$ for every $c \in C$ as follows:

(i) for all $a \in A$, $v_c(a) = a$,

(ii) if $t \in T$, then $v_c(\alpha(t)) = c(v_c(t))$,

(iii) if $\mathfrak{f}_i \in \mathscr{A}$, $\mathrm{ar}(\mathfrak{f}_i) = m$ and $t_1, \ldots, t_m \in T$, then

$$v_c(\mathfrak{f}_i(t_1, \ldots, t_m)) = f_i(v_c(t_1), \ldots, v_c(t_m)).$$

Suppose $r \in R$ and $r = t \to t'$. The realization of r is a partial function $\varrho_r : C \to C$ which assigns to a memory state $c \in C$ a new state $\varrho_r(c)$ (if defined) such that

$$\varrho_r(c)(a) = \begin{cases} v_c(t) & \text{if } a = v_c(t'), \\ \lambda(c(l)) & \text{if } a = l,\ a \neq v_c(t'), \\ c(a) & \text{if } a \neq l,\ a \neq v_c(t'), \end{cases}$$

where $\lambda : A \to A$ is a given function which changes the contents of the instruction counter l.

Let R be the instruction list of the SPC M. Defining a partial function $\varkappa$ from B onto R, we obtain a coding of instructions in M so that $\varkappa^{-1}(r)$ is the set of codes of r. $\varkappa$ will be called the *coding function* of M. Let

$$r_c = \varkappa(c(c(l))).$$

r_c is an instruction uniquely defined for each state c.

Now let π be defined as follows:

$$\pi(c) = \varrho_{r_c}(c)$$

for every $c \in C$. Hence $c \in D(\pi)$ iff $c(l) \in D(c)$ and $c(c(l)) \in D(\varkappa)$ and $c \in D(\varrho_{r_c})$.

Thus a stored program computer $M = \langle C, \pi \rangle$ is fully described by a sixtuple $M = \langle C, R, \varrho, \varkappa, \lambda, l \rangle$, where ϱ associates with every $r \in R$ its realization ϱ_r; the remaining components have been described above.

Let us now consider an instruction $r \in R$. We shall define the data and result regions of r in a state c. Assume $t \in T$. Then the *data region* $D_c(t)$ of the term t in a state c is defined as follows:

(i) $D_c(t) = \emptyset$ if $t \in A$,

(ii) $D_c(t) = D_c(t') \cup \{v_c(t')\}$ if $t = \alpha(t')$,

(iii) $D_c(t) = D_c(t_1) \cup \ldots \cup D_c(t_m)$ if $t = \mathfrak{f}_i(t_1, \ldots, t_m)$.

Then the data region of an instruction $r = t \to t'$ in a state $c \in C$ is the following set $D_c(r)$:

$$D_c(r) = D_c(t) \cup D_c(t').$$

The *result region* of r in a state c is simply the set

$$\mathrm{Re}_c(r) = \{v_c(t')\},$$

which is empty when $v_c(t')$ is not defined.

Let $M = \langle C, R, \varrho, \varkappa, \lambda, l \rangle$ be an SPC and $C \subset B^{[A]}$. A program in M is any partial function $\varphi : A - \{l\} \to R$ with a finite domain. Thus a program φ associates with every address $a \in D(\varphi)$ some instruction from the instruction list of the SPC. An element $(a, r) \in A - \{l\} \times R$ will be called a *labelled instruction*. Hence a program φ may be identified with a finite set of labelled instructions such that if $(a, r) \in \varphi$ and $(a, r_1) \in \varphi$ then $r = r_1$. We shall often consider programs as such sets (which are graphs of the corresponding partial functions).

We shall say that a program φ is stored in a state $c \in C$ (or c has the program φ stored) iff $\varkappa(c(a)) = \varphi(a)$ for each address $a \in D(\varphi)$. Thus φ is stored in c when for any labelled instruction $(a, r) \in \varphi$, a code of r constitutes the contents of the address a in the state c. We shall denote by C^φ the subset of C consisting of all states which have the program φ stored, i.e.

$$c \in C^\varphi \quad \text{iff } \varphi \text{ is stored in } c.$$

Let the *realization of a program* φ in M be a function $\varrho_\varphi : C \to C$ such that

$$\varrho_\varphi = f_M | C^\varphi,$$

where f_M is the result function in M. Then, for any $c \in C^\varphi$, $f_M(c)$ may be interpreted as the result of the program φ applied to c (which may be undefined when $c \notin D(f_M)$).

Let us now define some properties of programs. We shall say that a *program* φ *is open* iff there exists a state $c \in C^\varphi$ and $k > 0$ such that $\pi^k(c)(l) \notin D(\varphi)$. Therefore a program φ is open if during some computation starting with a state which has φ stored the instruction counter points to some address which is not a label in φ. A *program is closed* when it is not open.

We shall say that a *program* φ *is self-modifying* if during its computations it erases or changes some of its instructions. Strictly speaking, φ is self-modifying iff, for some $c \in C^\varphi$ and $k > 0$, $\pi^k(c) \notin C^\varphi$. A program which is not self-modifying will be called *fixed*.

Let φ be a program in M and assume $a \in D(\varphi)$. Then the pair $\langle \varphi, a \rangle$ will be called an *l-program* in M. If $p = \langle \varphi, a \rangle$ is an l-program, then C^p will denote the

set of all states $c \in C$ which have the l-program p stored, i.e.

$$C^p = \{c \in C^\varphi\colon c(l) = a\}.$$

In other words, an l-program is a program with the first instruction distinguished.

In the following sections we shall consider properties of computers and programs so defined.

2. Instruction list of SPC

We shall present here a reduction theorem for the number of instructions. This theorem states that the number of schemes of instructions in a stored program computer can often be essentialy reduced without decreasing the computing power of the computer.

Let M be a stored program computer with transition function π, instruction counter l, memory C, coding function $\varkappa$ and a set of instructions R. Let p be an l-program in M. We shall use the following notation:

$$\vec{C}^p = C^p \cup \pi(C^p) \cup \pi^2(C^p) \cup \ldots;$$

$$A^p = \{a \in A\colon (\exists c)_{C^p}(\exists k)_N(\pi^k(c)(l) = a)\};$$

$$P^p = \bigcup_{c \in \vec{C}^p} D_c(r_c);$$

$$D^p = P^p \cup A^p - \{l\};$$

$$I^p = \{r \in R\colon (\exists c')_C(\exists c)_{C^p}(\exists k)_N[(r = r_{c'}) \wedge (\pi^k(c) = c')]\}.$$

The set $\vec{C}^p$ is the closure of C^p in M. The set A^p is the set of all addresses with the following property: their contents (in a certain memory state) will be treated as an instruction code when the computer M starts a computation with a state in which the l-program p is stored. The set I^p is the set of all those instructions which are performed when the computer M starts a computation with a state in which the l-program p is stored. The set P^p is the sum of data regions of all instructions in the set I^p.

We shall consider only such l-programs p in M that $C^p \subset D(\pi_M)$.

By P_M we denote the set of all l-programs of the computer M and F_M will denote the set $\{\varrho^p\colon p \in P_M\}$.

We shall limit our considerations to machines fulfilling the following assumptions (i)–(iv):

(i) *Assumptions on memory* C:

(a) $A = B = N$;

(b) C is the set of all functions c from N into N such that $\{a\colon c(a) \neq 0\}$ is a finite set;

(c) An infinite partition of N is defined, i.e. a family of infinite sets $\{A_i\}_{i=1}^{\infty}$ such that $\bigcup_{i=1}^{\infty} A_i = N$ and $A_i \cap A_j = \emptyset$ for $i \neq j$. The elements of A_n will be denoted by a_i^n for $i \geqslant 0$.

(ii) *Assumptions on function* λ: for any $i, j \in N$ if $c(l) = a_i^j$, then $\lambda(c(l)) = a_{i+1}^j$.

By R^c we denote the set of instructions containing the following instructions (and only these instructions):

(1) $\alpha(b) \to a$, $\alpha(\alpha(b)) \to a$, $\alpha(a) \to \alpha(b)$ for $a, b \in N$;

(2) $b \to l$ for $b \in N$;

(3) $t \to l$ for such t that there exist $x, y, b \in N$ with the following property:

(a) for arbitrary $c \in C$

$$v_c(t) = \begin{cases} b & \text{if } c(x) = c(y), \\ \lambda(c(l)) & \text{if } c(x) \neq c(y). \end{cases}$$

If $x, y, b \in N$ and t is a term with property (3a), then the instruction $t \to l$ will be denoted by $[x, y, b]$.

(iii) *Assumptions on the instruction set* R:

(a) If $\{f_1, \ldots, f_k\}$ is the set of names of elementary operations, then

(1) $k \geqslant 3$;

(2) f_2 is the name of the function $f_2(n) = n+1$ for $n \in N$;

(3) f_3 is the name of the function $f_3(n) = n \dot{-} 1$ for $n \in N$;

(4) f_1 is the name of the empty function and $\mathrm{ar}(f_1) = 1$.

(b) For any $b, y, x \in N$ there exists a term t with the property (3a).

(c) $R^c \subset R$.

(d) $\{f_i(\alpha(a)) \to b\colon i = 2, 3;\ a, b \in N\} \subset R$.

(e) $f_1(\alpha(0)) \to l \in R$.

The instruction $f_1(\alpha(0)) \to l$ is denoted by STOP.

An *atomic instruction* is any instruction in the form

$$f(\alpha(a_1), \ldots, \alpha(a_k)) \to b,$$

where $a_1, \ldots, a_k, b \in N$, f is the name of an elementary operation and $\mathrm{ar}(f) = k$.

In the sequel some special kinds of l-programs will be used. An l-program $p = \langle \varphi, a \rangle$ in a stored program computer M will be called *regular* if the following conditions are satisfied:

(i) if $\varphi(x) = \mathrm{STOP}$, then $x \notin P_M^p$ (where $x \in D(\varphi)$);

(ii) if $c \in C_M^p \cap D(f_M)$, then $r_{c'} = \varphi(c'(l)) = \mathrm{STOP}$, where $c' = f_M(c)$.

A self-regenerating l-program (in M) is any l-program $p = \langle \varphi, a \rangle$ such that

$$C_M^p \subset D(f_M) \quad \text{and} \quad f_M(C_M^p) \subset C_M^\varphi.$$

Self-regenerating and regular l-programs are called *simple* l-programs in M. By Φ^p (or Φ_M^p) we denote the set of all simple l-programs in M.

Let $\{f_1, \ldots, f_k\}$ be the set of names of elementary operations of M and let $r = t \to t' \in R$. Let $a_0, a_1, \ldots, a_{k+3}$ be distinct even positive integers. The sequence obtained from r by substituting a_i for every f_k $(i = 1, \ldots, k)$, a_{k+1} for the symbol $($, a_{k+2} for), a_{k+3} for the symbol $\to$ and a_0 for α and $2m+1$ for any address m in r will be called the *number of* r. The mapping, thus defined, from R_M into N^+ will be denoted by $\boldsymbol{n}$ and the inverse mapping (with the domain $R(\boldsymbol{n})$) by $\boldsymbol{n}^{-1}$.

Now we are going to define some class of functions computable by a stored program computer M.

Let $g \in N^{+[N+1]}$. We shall call g simply *M-computable*, where M is a stored program computer, if for every $n \in N$ there exists an l-program $p \in \Phi_M^p$ such that

(1) $D_M^p \subset A_{n+2} \cup A_n \cup A_{n+1}$;

(2) $D(f_M) \supset C^p$;

(3) if $c \in C^p$, $c(a_0^n) = m$ and $\big(c(a_1^n), \ldots, c(a_m^n)\big) \in D(g)$, then $\big(f_M(c)(a_1^{n+2}), \ldots, f_M(c)(a_k^{n+2})\big) = g\big(c(a_1^n), \ldots, c(a_m^n)\big)$ and $f_M(c)(a_{k+1}^{n+2}) = 0$, where $k = f_M(c)(a_0^{n+2})$;

(4) if $c \in C^p$, $c(a_0^n) = m$ and $\big(c(a_1^n), \ldots, c(a_m^n)\big) \notin D(g)$, then $f_M(c)(a_{k+1}^{n+2}) = 1$, where $k = f_M(c)(a_0^{n+2})$;

(5) I_M^p has only a finite number of instructions containing names of elementary operations and these instructions are atomic.

The last assumption on computers is the following:

(iv) *Assumptions on coding functions and simply computable functions*:

(a) if M and M' are stored program computers, then the functions $\boldsymbol{n} \circ \varkappa_M$ and $\varkappa_M^{-1} \circ \boldsymbol{n}^{-1}$ are simply M'-computable;

(b) the function $f \in N^{+[N+1]}$ such that $D(f)$ is a set of sequences of length 1 and $f(a_i^j) = a_{i+1}^j$ for $i, j \geqslant 0$, is simply M-computable;

(c) $\varkappa_M$ is a bijection.

We can compare the realizations of l-programs using the relation S defined in the following way:

Let M, M' be stored program computers. We define a relation $S \subset F_M \times F_{M'}$ as follows: for any $\varrho \in F_M$, $\varrho' \in F_{M'}$

$\varrho S \varrho'$ iff there exists a one-to-one mapping α of N into N such that for every memory state c of the computer M there exists a memory state c'' of the computer M' such that

(a) $c'' > c_\alpha$, where $c_\alpha(x) = c\big(\alpha^{-1}(x)\big)$ for $x \in R(\alpha)$ and $D(c_\alpha) = R(\alpha)$;

(b) if $c \in D(\varrho)$, then $\varrho'(c'')(\alpha a) = \varrho(c)(a)$ for $a \in D(c)$;

(c) if $c \notin D\varrho$, then $c'' \notin D\varrho'$.

By $S[\varrho]$ we denote the set $\{g\colon \varrho S g\}$.

We say that the semantic of M is *expressible* in M' (and we denote this fact by $M < M'$) if for any i $(1 \leqslant i \leqslant k)$ there exist addresses $a_0^{(i)}, a_1^{(i)}, \ldots, a_{n_i}^{(i)}$ such that

$$a_j^{(i)} \neq a_{j'}^{(i)} \quad \text{for} \quad j \neq j' \ (0 \leqslant j \leqslant i,\ 0 \leqslant j' \leqslant i);$$

$$\operatorname{ar}(f_i) = n_i \quad \text{and} \quad f_i\big(\alpha(a_1^{(i)}), \ldots, \alpha\big(a_{n_i}^{(i)}\big)\big) \to a_0^{(i)} \in R_{M'},$$

where $\{f_1, \ldots, f_k\}$ is the set of elementary operations of M.

If $P \subset P_{M'}$, then we say that P is of *M-finite type* if the set

$$\bigcup_{p \in P} I_{M'}^p - R_M^c$$

is finite and includes only atomic instructions of computer M.

If $P \subset P_{M'}$, then by F_P we denote the set

$$\{\varrho_{M'}^p\colon p \in P\}.$$

The second relation which we define in the family of stored program computers is the relation of inclusion.

Let M and M' be stored program computers. We say that M is included in M', which is denoted by $M \prec M'$, iff there exists a set of M-finite type programs $P \subset P_{M'}$ such that

$$(\forall \varrho)_{F_M}(S[\varrho] \cap F_P \neq \emptyset).$$

We have the following theorem.

THEOREM 2.1. *Let M, M' be stored program computers fulfilling assumptions* (i)–(iv). *If $M < M'$, then $M \prec M'$.*

Proof. Let $\alpha_k\colon N \to N$ and $\alpha_k(n) = a_{2n+1}^{k+1}$ for $n = 0, 1, 2, \ldots$ If $c \in C$, then by c_{α_k} we denote a function

$$c_{\alpha_k}\colon \{a_1^{k+1}, \ldots, a_{2n+1}^{k+1}, \ldots\} \to N$$

such that $c_{\alpha_k}(x) = c\big(\alpha_k^{-1}(x)\big)$ for $x \in R(\alpha_k)$. By $Z(c, \alpha_k)$ we denote the set (see Fig. 1)

$$\big\{c'' \in N^N\colon c''|D(c_{\alpha_k}) = c_{\alpha_k} \wedge (\forall n)_N\big(c''(a_{2n}^{k+1}) = n\big)\big\}.$$

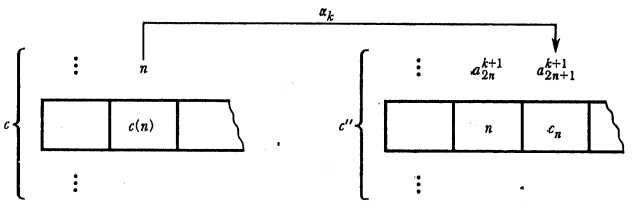


Fig. 1

Let $\{f_1, \ldots, f_n\}$ be the set of elementary operations of M and, for $i = 1, \ldots, n$, let $a_0^{(i)}, \ldots, a_{n_i}^{(i)}$ be distinct addresses such that

$$f_i\big(\alpha(a_1^{(i)}), \ldots, \alpha(a_{n_i}^{(i)})\big) \to a_0^i \text{ are in } R_{M'}.$$

Let k be a nonnegative integer such that the addresses mentioned above and $l_{M'}$ are in $A_1 \cup \ldots \cup A_k$.

It is possible to construct an l-program q which has the following properties:

(i) $\{q\}$ is an M-finite type set of programs in M';

(ii) for an arbitrary l-program p in M and $c \in C^p$

$$c \in D(f_M) \equiv (\forall c'')_{C^q \cap Z(c, \alpha_k)}\big(c'' \in D(f_M)\big);$$

(iii) if $c \in C^p \cap D(f_M)$ and $c'' \in C^q \cap Z(c, \alpha_k)$, then for $a \in N$

$$f_{M'}(c'')\big(\alpha(a)\big) = f_M(c)(a).$$

From (ii)–(iii) it follows that if p is an l-program in M, then

$$\varrho^p S \varrho^q.$$

Hence $(\forall \varrho)_{F_M}(\varrho^q \in S[\varrho])$ and $(\forall \varrho)_{F_M}(S[\varrho] \cap \{\varrho^q\} \neq \emptyset)$.

From the above considerations we infer that the construction of q with properties (i)–(iii) ends the proof. ■

Now we are going to present the construction of such an l-program.

If $c \in C$ and (1) $c(a_{2n}^{k+1}) = n$ for $n = 0, 1, \ldots$, then by c_h we denote a function from N in N which is defined as follows:

$$c_h(n) = c(a_{2n+1}^{k+1}) \quad \text{for} \quad n = 0, 1, \ldots$$

If φ is a program, then by $\varphi(a)$ we denote the program which is obtained from φ by substituting the instruction $a \to l$ for any instruction STOP in φ, i.e.

$$\varphi(a)(x) = \begin{cases} x & \text{if } x \neq \text{STOP}, \\ a \to l & \text{otherwise}. \end{cases}$$

First we shall construct an l-program $q_1 = \langle \varphi_1, a_0^{k+100} \rangle$ which has the following property: for any $c \in C^{q_1}$, if (1) is satisfied, then $f_M(c)(a_1^{k+10})$ is the code of the instruction of the computer M executed in the state c_h. We define φ_1 as follows:

$$\begin{aligned}
\varphi_1 = \{ & \langle a_0^{k+100}, \alpha(a_{2l_M+1}^{k+1}) \to a_0^{k+3} \rangle, \\
& \langle a_1^{k+100}, a_0^{k+1} \to a_1^{k+5} \rangle, \\
& \langle a_2^{k+100}, 1 \to a_0^{k+5} \rangle, \\
& \langle a_3^{k+100}, \alpha(\alpha(a_1^{k+5})) \to a_0^{k+4} \rangle, \\
& \langle a_4^{k+100}, [a_0^{k+3}, a_0^{k+4}, a_{13}^{k+100}] \rangle, \\
& \langle a_5^{k+100}, \alpha(a_1^{k+5}) \to a_1^{k+101} \rangle, \\
& \langle a_6^{k+100}, 1 \to a_0^{k+101} \rangle, \\
& \langle a_7^{k+100}, a_{k+102} \to l \rangle \} \cup \\
& \cup \varphi_{k+102}(a_8^{k+100}) \cup \\
& \cup \{ \langle a_8^{k+100}, \alpha(a_1^{k+103}) \to a_1^{k+104} \rangle, \\
& \langle a_9^{k+100}, 1 \to a_0^{k+104} \rangle, \\
& \langle a_{10}^{k+100}, a_{k+105} \to l \rangle \} \cup \\
& \cup \varphi_{k+105}(a_{11}^{k+100}) \cup \\
& \cup \{ \langle a_{11}^{k+100}, \alpha(a_1^{k+106}) \to a_1^{k+5} \rangle, \\
& \langle a_{12}^{k+100}, a_3^{k+100} \to l \rangle, \\
& \langle a_{13}^{k+100}, \alpha(a_1^{k+5}) \to a_1^{k+107} \rangle, \\
& \langle a_{14}^{k+100}, 1 \to a_0^{k+107} \rangle, \\
& \langle a_{15}^{k+100}, a_{k+108} \to l \rangle \} \cup \\
& \cup \varphi_{k+108}(a_{16}^{k+100}) \cup \\
& \cup \{ \langle a_{16}^{k+100}, \alpha(\alpha(a_1^{k+109})) \to a_1^{k+10} \rangle, \\
& \langle a_{17}^{k+100}, \text{STOP} \rangle \},
\end{aligned}$$

where the function $f(a_j^{j'}) = a_{j+1}^{j'}$ $(j, j' = 0, 1, \ldots)$ is simply M'-computable by $q_{k+i} = \langle \varphi_{k+i}, a_{k+i} \rangle$ and

$$D^{q_{k+i}} \subset A_{k+i-1} \cup A_{k+i} \cup A_{k+i+1} \quad (i = 1, 2, \ldots).$$

By the assumptions of Theorem 2.1 there exists a simple l-program in M', $q_2 = \langle \varphi_2, a_2 \rangle$ such that $D^{q_2} \subset A_{k+10} \cup A_{k+11} \cup A_{k+12}$ and the function $\boldsymbol{n} \circ \varkappa_{\overline{M}}^{-1}$ is simply M'-computable by q_2. Let

$$\begin{aligned}
\varphi_3 = (\varphi_1 - \{ \langle a_{17}^{k+100}, \text{STOP} \rangle \}) \cup \{ & \langle a_{17}^{k+100}, 1 \to a_0^{k+10} \rangle, \\
& \langle a_{18}^{k+100}, a_2 \to l_{M'} \rangle \} \cup \varphi_2.
\end{aligned}$$

An l-program $q_3 = \langle \varphi_3, a_0^{k+100} \rangle$ in M' has the following property:

$$\text{if } c \in C^{q_3}, \text{ then } c \in D(f_{M'}) \text{ and } f_{M'}(c)(a_1^{k+12}) = \boldsymbol{n}\big(\varkappa_M(c_h^2(l))\big).$$

It is possible to construct an l-program $q_4 = \langle \varphi_4, a_0 \rangle$ with the following properties:

(a) there exists an $a \in D(\varphi_4)$ such that if the state c fulfils the following condition:

(2) $c \in C^{q_4}$ and (1) holds and for some $n, b_1, \ldots, b_n$ we have $c(a_0^{k+12}) = n$, $c(a_1^{k+12}) = b_1, \ldots, c(a_n^{k+12}) = b_n$ and $b_1, \ldots, b_n$ is the number of the instruction $r = t \to t' \in R_M$,

then

(a1) $f_{M'}(c)(l) = a \equiv c_h \in D(\pi_M)$;

(a2) $a \notin P_M^{q_4}$;

(b) if condition (2) holds and $c_h \in D(\pi_M)$, then

$$v_{c_h}(t) = f_{M'}(c)(a_0^{k+12});$$
$$v_{c_h}(t') = f_{M'}(c)(a_1^{k+12});$$

(c) $f_{M'}(c)|N - (A_{k+12} \cup \{l\}) = c|N - (A_{k+12} \cup \{l\})$ if (2) holds.

In the same way as in the first step of the proof one can construct a simple l-program $q_5 = \langle \varphi_5, a_5 \rangle$ such that: if $c \in C^{q_5}$ and $c(a_1^{k+12}) = n$, then $f_{M'}(c)(a_1^{k+30}) = a_{2m+1}^{k+1}$, where m is such a number that

$$c(a_{2m}^{k+1}) = n \quad \text{and} \quad f_{M'}(c)|N - (\{l\} \cup A_{30+k}) = c|N - (A_{30+k} \cup \{l\}).$$

Let

$$\varphi_6 = \varphi_4(a/a_5) \cup \varphi_5(a_0^{k+31}) \cup \{ \langle a_0^{k+31}, \alpha(a_0^{k+12}) \to \alpha(a_1^{k+30}) \rangle, \langle a_1^{k+31}, a_0^{k+100} \to l \rangle \},$$

where

$$\varphi_4(a/a_5)(x) = \begin{cases} \varphi_4(x) & \text{if } x \neq a, \\ a_5 \to l & \text{otherwise}. \end{cases}$$

It is possible to prove that an l-program

$$q = \langle \varphi_3 \cup \varphi_6, a_0^{k+100} \rangle$$

has mentioned properties (i)–(iii).(2) ■

(2) One can construct l-programs φ_3, φ_4, φ_5 with pairwise disjoint domains.

3. Algebraic complexity of programs

In this section we associate with each program in an SPC a pair of iterative systems which represent the computations of the program and the schemes of these computations. Thus programs may be compared through their iterative systems.

Let $M = \langle C, \pi\rangle$ be an SPC and φ a program in M. As usual, $\overline{C^\varphi}$ denotes the closure of the set C^φ in M, i.e.

$$\overline{C^\varphi} = \bigcup_{i=0}^{\infty} \pi^i(C^\varphi).$$

Thus $c \in \overline{C^\varphi}$ iff c can be reached through some iteration of π from a state which has the program φ stored.

Let $M(\varphi) = \langle \overline{C^\varphi}, \pi\rangle$. This subsystem of M will be called the *system of the program* φ. Its computations are all computations in M induced by the program φ.

Every computation $\bar{c}_0 = (c_0, c_1, \ldots)$ in M determines a unique sequence of labelled instructions of the following form:

$$S(\bar{c}_0) = \big((c_0(l), c_0^2(l)), (c_1(l), c_1^2(l)), \ldots\big),$$

where l is the instruction counter of M. The sequence $S(\bar{c}_0)$ will be called the *scheme of the computation* $\bar{c}_0$. The scheme $S(\bar{c}_0)$ is the sequence of all labelled instructions performed during the computation $\bar{c}_0$.

Suppose $M(\varphi) = \langle \overline{C^\varphi}, \pi\rangle$ is the system of a program φ. We define a relation R_M^* in the set $\overline{C^\varphi} \cap \big(D(\pi) \cup R(\pi)\big)$ of all active states of $M(\varphi)$ as follows:

(1) $(c, c_1) \in R_M^*$ iff $[c, c_1 \in D(\pi)\ \&\ S(\bar{c}) = S(\bar{c}_1)] \vee$
$\vee [c, c_1 \notin D(\pi)\ \&\ (\exists c', c_1')\ (\pi(c') = c\ \&\ \pi(c_1') = c_1\ \&\ S(\bar{c}') = S(\bar{c}_1'))]$.

Thus two states are in the relation R_M^* iff they are initial (or final) states of computations with equal schemes. It is easily verified that R_M^* is a congruence in the subsystem $\overline{C^\varphi} \cap \big(D(\pi) \cup R(\pi)\big)$ of the system $M(\varphi)$. Therefore we may consider the quotient system $M^*(\varphi) = \langle \overline{C^\varphi} \cap \big(D(\pi) \cup R(\pi)\big)/R_M^*, \pi^*\rangle$ which will be called the *schematic system of* φ, since there is a 1-1 correspondence between the states of $M^*(\varphi)$ which lie in the domain of π^* and the schemes of computations in $M(\varphi)$.

Thus any program φ in M may be described through two iterative systems associated with it: the system $M(\varphi)$, which represents all the computations induced by φ, and the schematic system $M^*(\varphi)$, which represents all the schemes of computations in $M(\varphi)$. We shall prove that these descriptions cover the whole class of non-empty at most countable iterative systems, i.e. each such system with a given congruence describes some program in a certain SPC.

THEOREM 3.1. *For any non-empty iterative system $N = \langle X, \pi\rangle$ and any congruence ϱ in N such that X/ϱ is at most countable, there exists a stored program computer $M = \langle C, \pi\rangle$ and a program φ in M such that there is an isomorphism h of N onto $M(\varphi)$ which satisfies the following: for any $x, x_1 \in D(\pi)$, $(x, x_1) \in \varrho$ iff $\big(h(x), h(x_1)\big) \in R_M^*$.*

Proof. Given an iterative system $N = \langle X, \pi\rangle$ and a congruence ϱ in N, define an SPC $M = \langle C, \pi_1\rangle$ as follows:

$A = X/\varrho \cup \{0, 1\}$,

$B = X/\varrho \cup X \cup \{0, 1\}$ and $\{0, 1\} \cap (X/\varrho \cup X) = \emptyset$,

$C = \big\{c \in B^A\colon c(1) \in X\ \&\ c(0) = [c(1)]_\varrho\ \&\ (\forall a \in X/\varrho)\,(a \in D(\pi^*) \Rightarrow c(a) = 1$
$\&\ a \notin D(\pi^*) \Rightarrow c(a) = 0)\big\}$,

where π^* is the transition function in the quotient system N/ϱ.

Observe that memory states in C are total functions and for any $c, c_1 \in C$,

(2) $$c|X/\varrho = c_1|X/\varrho.$$

Associate with M one unary operation π. Let $\mathscr{A} = \{\mathfrak{f}, \alpha, (,), '\}$ be an alphabet over which the terms are built. Take an instruction list R consisting of two instructions only:

$$R = \big\{\mathfrak{f}(\alpha(1)) \to 1, r_1\big\},$$

where r_1 is any instruction (e.g. $r_1 = \alpha(1) \to 1$, an instruction which changes at most the instruction counter) and $\mathfrak{f}$ is a name for π. Let λ, the function which changes the instruction counter, be any function from A into A such that

$$\lambda|X/\varrho = \pi^*.$$

The address 0 is the instruction counter of M.

Finally, let $\varkappa\colon B \to R$ be a partial function such that $D(\varkappa) = X \cup \{1\}$ and

$$\varkappa(1) = r,$$
$$\varkappa(x) = r_1 \quad \text{for all } x \in X.$$

Now for any $x \in X$ let c_x be a state in C such that $c_x(0) = [x]_\varrho$ and $c_x(1) = x$ (by (2) this defines c_x uniquely). It follows from the definition of C that the mapping

$$h(x) = c_x$$

is a mapping of X onto C. Assume $x \in D(\pi)$. Then $[x] \in D(\pi^*)$ and $c_x\big(c_x(0)\big) = c_x([x]) = 1$, which ensures that $\varkappa$ is defined on $c_x\big(c_x(0)\big)$. Therefore $c_x \in D(\pi_1)$. Similarly, $x \notin D(\pi)$ implies $c_x \notin D(\pi_1)$, since $\varkappa\big(c_x\big(c_x(0)\big)\big) = \varkappa(0)$ is not defined. Thus

(3) $$x \in D(\pi) \quad \text{if and only if} \quad c_x \in D(\pi_1).$$

Suppose $c_x \in D(\pi_1)$. Then $r_{c_x} = r$ and, by the definition of a realization of an instruction, $\varrho_r(c_x)$ is a state such that

$$\varrho_r(c_x)(a) = \begin{cases} \pi\big(c_x(1)\big) & \text{if } a = 1, \\ \pi^*\big(c_x(0)\big) & \text{if } a = 0, \\ c_x(a) & \text{if } 0 \neq a \neq 1. \end{cases}$$

Thus $\varrho_r(c_x)(0) = \pi^*\big(c_x(0)\big) = \pi^*([x]) = [\pi(x)]$ and $\varrho_r(c_x)(1) = \pi\big(c_x(1)\big) = \pi(x)$. Hence

(4) $$\pi_1(c_x) = \varrho_r(c_x) = c_{\pi(x)}.$$

It follows from (3) and (4) that

(5) $$\text{if } x \in R(f_N), \text{ then } c_x \in R(f_M).$$

Therefore, by (4) and (5) the mapping h is a homomorphism of N onto M. Since it is obviously one-to-one, and by (3), it is an isomorphism. It is an obvious fact that by the construction of M, if we take $\varphi = \{(1, r_1)\}$, then $M = M(\varphi)$. Thus h is an isomorphism of N onto $M(\varphi)$.

To prove that h is the required isomorphism, assume $x, x_1 \in D(\pi)$ and $(x, x_1) \in \varrho$. By (3), $c_x, c_{x_1} \in D(\pi_1)$. Observe that $c_x(0) = [x] = [x_1] = c_{x_1}(0)$ and $c_x(c_x(0)) = c_x([x]) = 1 = c_{x_1}(c_{x_1}(0))$. It may be proved by induction that these equalities hold for any iteration of π_1, as long as it is defined (in that case it must be defined for both c_x and c_{x_1}), i.e.

$$\pi_1^k(c_x)(0) = \pi_1^k(c_{x_1})(0)$$

and

$$\pi_1^k(c_x)\left(\pi_1^k(c_x)(0)\right) = \pi_1^k(c_{x_1})\left(\pi_1^k(c_{x_1})(0)\right).$$

This proves that

$$S(\bar{c}_x) = S(\bar{c}_{x_1})$$

and therefore $(c_x, c_{x_1}) \in R_M^*$.

Conversely, assume $x, x_1 \in D(\pi)$ and $(c_x, c_{x_1}) \in R_M^*$. Since c_x and c_{x_1} belong to $D(\pi_1)$ by (3), we have $S(\bar{c}_x) = S(\bar{c}_{x_1})$. In particular, $c_x(0) = c_{x_1}(0)$, which is equivalent to $[x] = [x_1]$. Hence $(x, x_1) \in \varrho$, which completes the proof. ■

Using the above theorem, we may reduce the investigation of some properties of programs (which can be expressed in terms of iterative systems) to the investigation of such systems and congruences in them. In the sequel we shall omit the assumptions of iterative systems being at most countable. All the results hold for such systems and all the constructions used render at most countable systems if performed on at most countable families of at most countable systems.

Let us consider the following relation $\leftrightarrow$ in the class of all iterative systems: $M \leftrightarrow M_1$ iff there exist homomorphisms of M into M_1 and of M_1 into M.

This is easily seen to be an equivalence relation (superposition of homomorphisms being a homomorphism again). Homomorphisms being mappings which preserve some general algebraic structure of systems, $M \leftrightarrow M_1$ means that two systems have a similar structure or, in other words, have similar algebraic complexity. Therefore the equivalence classes of $\leftrightarrow$ will be called *complexity classes.*

We shall say that programs φ in an SPC M and φ_1 in an SPC M_1 are

—*weakly similar* iff $M(\varphi) \leftrightarrow M_1(\varphi_1)$,

—*similar* iff moreover $M^*(\varphi) \leftrightarrow M_1^*(\varphi_1)$.

Thus two programs are weakly similar when they have "similar" computations and they are similar when they also have "similar" schemes of computations.

In the family of all complexity classes we introduce an ordering relation $\rightarrow$: $[M] \rightarrow [M_1]$ iff there exists a homomorphism of M into M_1. This relation clearly does not depend on the choice of systems from the classes $[M]$ and $[M_1]$.

Suppose $\{M_i\}_{i\in I}$ is a family of iterative systems, $M_i = \langle X_i, \pi_i\rangle$. The *disjoint sum* of this family is a system

$$\mathop{S}_{i\in I} M_i = \langle X, \pi\rangle$$

such that

$$X = \bigcup_{i\in I} (X_i \times \{i\}), \qquad D(\pi) = \bigcup_{i\in I} \left(D(\pi_i) \times \{i\}\right)$$

(i.e. $(x, i) \in D(\pi)$ iff $x \in D(\pi_i)$), and $\pi\big((x, i)\big) = \big(\pi_i(x), i\big)$ for any $(x, i) \in D(\pi)$.

In particular, when the sets X_i are pairwise disjoint, the system $\langle \bigcup_{i\in I} X_i, \bigcup_{i\in I} \pi_i\rangle$ is (up to isomorphism) the disjoint sum of the family of systems $\{M_i\}_{i\in I}$.

For any $i \in I$, the mapping $h_i: X_i \to X$ such that $h_i(x) = (x, i)$ is a homomorphism of M_i into $\mathop{S}_{i\in I} M_i$. Moreover, if $\{g_i\}_{i\in I}$ is a family of homomorphisms with $g_i: X_i \to X'$ for some system $M' = \langle X', \pi'\rangle$, then a mapping $g: X \to X'$ such that $g\big((x, i)\big) = g_i(x)$ is a homomorphism of $\mathop{S}_{i\in I} M_i$ into M'. This proves the following theorem:

THEOREM 3.2. *Let $\{M_i\}_{i\in I}$ be a family of iterative systems. There exists a least upper bound (with respect to $\rightarrow$) for the family $\{[M_i]\}_{i\in I}$ of their complexity classes.* ■

Thus complexity classes with $\rightarrow$ form a complete upper semilattice.

Observe that each iterative system M is a disjoint sum of three subsystems (some of them empty, perhaps): the first contains all the states which do not belong to any infinite computation of M, the second contains all the states which appear in some infinite non-cyclic computation of M and the third consists of all the states which appear in some cyclic computation of M. Thus in order to describe in more detail the complexity classes and their ordering we shall distinguish three kinds of iterative systems: finite, infinite and cyclic.

Finite systems. A non-empty system $M = \langle X, \pi\rangle$ will be called *finite* iff $D(\pi) = D(f_M)$, i.e. when every computation in M is finite. Observe that a finite system need not have a finite set of states. Let $\mathcal{FS}$ be the class of all finite systems.

Let H_M be the function defined for a system M in Section I.4. If M is a finite system, then H_M is defined on all its active states. The following lemma holds:

LEMMA 3.1. *If $m \neq n$, then $H_M^{-1}(m) \cap H_M^{-1}(n) = \emptyset$.*

Proof. Assume $m > n$ and let $x \in H_M^{-1}(m) \cap H_M^{-1}(n)$. Then by the definition of H_M, $\pi^n(x) \in R(f_M)$ and $\pi^{m-n}(\pi^n(x))$ is defined and belongs to $R(f_M)$, which is a contradiction (by Lemma I.2.1, $R(f_M) \cap D(\pi) = \emptyset$). ■

Let $M = \langle X, \pi\rangle$ be an arbitrary iterative system. An infinite sequence $(x_0, x_1, \ldots)$ of elements of X is said to be a *computation extended on the left* (c.e.l.) of M iff for every $i > 0$ the sequence $(x_i, \ldots, x_0)$ is a computation in M (which is equivalent to $x_0 \in R(f_M)$ and $\pi(x_{i+1}) = x_i$ for all $i \geqslant 0$).

THEOREM 3.3. *If a system $M_1 = \langle X_1, \pi_1\rangle$ has a* c.e.l., *then for any finite system $M = \langle X, \pi\rangle$ there exists a homomorphism of M into M_1.*

Proof. Let $(d_0, d_1, \ldots)$ be a c.e.l. in M_1 and assume $M \in \mathscr{FS}$. Define $h: X \to X_1$ as follows:

for all $x \in X - D(\pi)$, $h(x) = d_0$,

for $x \in D(\pi)$, $h(x) = d_i$ iff $x \in H_M^{-1}(i)$.

By Lemma 3.1, h is well defined on X. It is easy to prove that it is a homomorphism of M into M_1. ■

THEOREM 3.4. *Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be iterative systems, $M \in \mathscr{FS}$. If M has a* c.e.l., *then there exists a homomorphism of M into M_1 if and only if M_1 has a* c.e.l.

Proof. The "if" part follows from the previous theorem. On the other hand, if $(d_0, d_1, \ldots)$ is a c.e.l. in M and h is a homomorphism of M into M_1, then by Theorem I.3.1, $\big(h(d_i), \ldots, h(d_0)\big)$ is a computation in M_1 for any $i > 0$, which proves that the sequence $\big(h(d_0), h(d_1), \ldots\big)$ is a c.e.l. in M_1. ■

From the above theorems we infer the following characterization of systems with c.e.l.:

THEOREM 3.5. *A system M_1 has a* c.e.l. *if and only if for any system $M \in \mathscr{FS}$ there exists a homomorphism of M into M_1.* ■

It follows from Theorems 3.3 and 3.4 that all finite systems with c.e.l. belong to one equivalence class of $\leftrightarrow$ and there are no other systems in that class. Let this class be denoted by CEL. The following is a consequence of Theorem 3.5:

COROLLARY 3.1. CEL *is the greatest element in the family of all complexity classes of finite systems.* ■

Let us now turn to finite systems without c.e.l. The class of such systems will be denoted by $\mathscr{FS}_0$. Thus $\mathscr{FS}_0 = \mathscr{FS}$-CEL.

LEMMA 3.2. *Let $M = \langle X, \pi\rangle \in \mathscr{FS}_0$. Then $X_0 = X - R(\pi)$ is the least generating set for M.*

Proof. Suppose X_0 is not a generating set for M. Then there exists a state $x \in R(\pi)$ such that for any $y \in X$, if $\pi^k(y) = x$, for some $k \geqslant 0$, then $y \in R(\pi)$. Observe that since $M \in \mathscr{FS}_0$, $x \in D(f_M) \cup R(f_M)$ and $H_M(x)$ is defined. Therefore we can choose a sequence

$$\pi^{H(x)}(x), \pi^{H(x)-1}(x), \ldots, \pi(x), x, x_1, x_2, \ldots$$

such that for any $i \geqslant 1$, $\pi^i(x_i) = x$. It is easy to check that this sequence is a c.e.l. in M, which contradicts $M \in \mathscr{FS}_0$. Thus X_0 must be a generating set.

Moreover, suppose $A \subset X$ and $X_0 - A \neq \emptyset$. If $x \in X_0 - A$, then $x \notin R(\pi)$. Thus x cannot be reached by any iteration of π from a state in A, which proves that A is not a generating set for M. Hence X_0 is the least generating set. ■

Assume that $M \in \mathscr{FS}_0$, $M = \langle X, \pi\rangle$ and M is connected. Define a function o_M which assigns ordinals to states of M in the following way:

(i) for every $x \in X - R(\pi)$, $o_M(x) = 1$,

(ii) for every $x \in R(\pi)$, $o_M(x)$ is the least ordinal greater than any ordinal in the set $\{o_M(y)\colon (\exists k > 0)(\pi^k(y) = x)\}$.

It follows from Lemma 3.2 that o_M is well defined on the whole set X.

Since M is connected, there exists exactly one state $x_0 \in X$ such that $x_0 \notin D(\pi)$: if $D(\pi) \neq \emptyset$, then $\{x_0\} = R(f_M)$; otherwise $\{x_0\} = X$. Thus we may define uniquely the type of M as the ordinal $o_M(x_0)$. Let $o(M)$ denote the type of M. It may be proved that every non-zero (and countable, if we restrict ourselves to countable systems) ordinal is the type of some connected system in $\mathscr{FS}_0$. Also the following lemma holds (the easy proof is omitted):

LEMMA 3.3. *Let $M = \langle X, \pi\rangle \in \mathscr{FS}_0$ be a connected system. If $\alpha = o(M)$, then for any non-zero ordinal $\beta \leqslant \alpha$ there exists a state $x \in X$ such that $o_M(x) = \beta$.* ■

The next lemma is an immediate consequence of a result by M. Novotny ([17], [18]):

LEMMA 3.4. *If $M = \langle X, \pi\rangle$ and M_1 are connected systems in $\mathscr{FS}_0$ and h is a homomorphism of M into M_1, then for any $x \in X$, $o_M(x) \leqslant o_{M_1}\big(h(x)\big)$.* ■

Now we can prove the necessary and sufficient condition for the existence of homomorphisms for two connected systems in $\mathscr{FS}_0$.

THEOREM 3.6. *Let α be any non-zero ordinal. If $M' = \langle X', \pi'\rangle$ is in $\mathscr{FS}_0$ and is a connected system of type α, then for any connected system $M = \langle X, \pi\rangle \in \mathscr{FS}_0$ there exists a homomorphism of M into M' if and only if $o(M) \leqslant \alpha$.*

Proof. The necessity of this condition follows from Lemma 3.4 by the definition of a homomorphism.

To prove sufficiency, assume $\alpha = o(M') = 1$. This is equivalent to $D(\pi') = \emptyset$ and if $o(M) \leqslant \alpha$, then also $D(\pi) = \emptyset$. Therefore the only mapping $h: X \to X'$ (X and X' are one-element sets) is obviously a homomorphism.

Suppose now $\alpha > 1$ and assume that the theorem holds for all ordinals γ such that $1 \leqslant \gamma < \alpha$.

Let $\alpha = \beta + 1$ for some $\beta \geqslant 1$ and suppose that $M = \langle X, \pi\rangle$ is a connected system in $\mathscr{FS}_0$ with $o(M) = \delta \leqslant \alpha$. Observe that if $H_M^{-1}(1) = \emptyset$, then $o(M) = 1$, $D(\pi) = \emptyset$ and again any mapping $h: X \to X'$ is a homomorphism. Therefore we can assume $H_M^{-1}(1) \neq \emptyset$.

For any state $x \in H_M^{-1}(1)$ define a system $M(x) = \langle U(x), \pi_x\rangle$, where

$$U(x) = \big\{y \in X\colon (\exists k \geqslant 0)\big(\pi^k(y) = x\big)\big\} \tag{6}$$

and

$$\pi_x = \pi \big| \big(U(x) - \{x\}\big).$$

Any such system is again connected, belongs to $\mathscr{FS}_0$ and $R(f_{M(x)}) = \{x\}$. Besides, $o\big(M(x)\big) = o_M(x) < \delta$.

Similarly,[3] for any $z \in H_{M'}^{-1}(1)$ define a system $M'(z) = \langle U(z), \pi'_z\rangle$, where $U(z)$ is a subset of X' defined as in (6) and $\pi'_z = \pi' \big| \big(U(z) - \{z\}\big)$. Such systems are also connected, belong to $\mathscr{FS}_0$ and $R(f_{M'(z)}) = \{z\}$. Moreover, $o\big(M'(z)\big) = o_{M'}(z) < \alpha$. By Lemma 3.3, there exists a state $z_0 \in H_{M'}^{-1}(1)$ such that $o\big(M'(z_0)\big) = o_{M'}(z_0) = \beta$. Since for every $x \in H_M^{-1}(1)$ we have $o\big(M(x)\big) < \delta \leqslant \alpha$, therefore $o\big(M(x)\big) \leqslant \beta$

[3] Observe that $H_{M'}^{-1}(1) \neq \emptyset$, since $o(M') = \alpha > 1$.

and by our assumption for any system $M(x)$ there exists a homomorphism h_x of $M(x)$ into $M'(z_0)$, which must satisfy $h_x(x) = z_0$. Hence we can define a mapping h: $X \to X'$ such that for any $a \in X$,

$$h(a) = \begin{cases} h_x(a) & \text{if } a \in U(x),\ x \in H_M^{-1}(1), \\ b & \text{if } a \in R(f_M), \end{cases} \tag{7}$$

where $\{b\} = R(f_{M'})$. It may easily be verified that h is well defined (the sets $U(x)$ for $x \in H_M^{-1}(1)$ are a partition of $X - R(f_M)$) and a homomorphism of M into M'.

Assume now that α is a limit ordinal and $M = \langle X, \pi\rangle \in \mathscr{FS}_0$ is connected and $o(M) = \delta \leqslant \alpha$. As before, we may assume $H_M^{-1}(1) \neq \emptyset$. Define the systems $M(x)$ for $x \in H_M^{-1}(1)$ and $M'(z)$ for $z \in H_{M'}^{-1}(1)$ as above. We then have $o\big(M(x)\big) < \delta \leqslant \alpha$, $o\big(M'(z)\big) < \alpha$ and for any ordinal $\gamma < \alpha$ there exists a state $z \in H_{M'}^{-1}(1)$ such that $\gamma < o\big(M'(z)\big) < \alpha$ by Lemma 3.3. Thus for every $x \in H_M^{-1}(1)$ there exists a $z_x \in H_{M'}^{-1}(1)$ such that $o\big(M(x)\big) \leqslant o\big(M'(z_x)\big)$. Therefore, by assumption, there exists a homomorphism h_x of $M(x)$ into $M'(z_x)$ with $h_x(x) = z_x$. Again define a mapping h as in (7). It is a well-defined homomorphism of M into M'. In consequence, the theorem holds for any ordinal $\alpha \geqslant 1$. ■

Every system being a disjoint sum of its m.c.-subsystems, we obtain by Theorem I.3.2 and the previous theorem the following

COROLLARY 3.2. *Let M, $M_1 \in \mathscr{FS}_0$. There exists a homomorphism of M into M_1 if and only if for every* m.c.*-subsystem S in M there exists an* m.c.*-subsystem S_1 in M_1 such that $o(S) \leqslant o(S_1)$.* ■

Assume now that $M \in \mathscr{FS}_0$. Suppose there is a greatest ordinal α in the set $\mathscr{O}(M) = \{o(S)\colon S$ is an m.c.-subsystem of $M\}$. It follows from Corollary 3.2 that for any $M_1 \in \mathscr{FS}_0$ there exists a homomorphism of M into M_1 iff for some m.c.-subsystem S_1 in M_1 $o(S_1) \geqslant \alpha$. In consequence, $M \leftrightarrow M_1$ iff α is the greatest ordinal in the set $\mathscr{O}(M_1) = \{o(S_1)\colon S_1$ is an m.c.-subsystem of $M_1\}$.

Thus for every ordinal $\alpha \geqslant 1$ we may define a class F_α of finite systems without c.e.l. such that

$$M \in F_\alpha \text{ iff } \alpha \text{ is the greatest ordinal in } \mathscr{O}(M).$$

On the other hand, assume that there is no greatest ordinal in the set $\mathscr{O}(M)$ and let $\alpha = \sup \mathscr{O}(M)$. Then for any m.c.-subsystem S in M we have $o(S) < \alpha$ and if $\gamma < \alpha$ then, for some m.c.-subsystem S' in M, $\gamma < o(S') < \alpha$. Therefore there exists a homomorphism of $M_1 \in \mathscr{FS}_0$ into M iff, for every m.c.-subsystem S_1 of M_1, $o(S_1) < \alpha$. On the other hand, there exists a homomorphism of M into M_1 iff for every m.c.-subsystem S of M there is an m.c.-subsystem S_1 in M_1 such that $o(S) \leqslant o(S_1)$. Thus $M \leftrightarrow M_1$ iff $\alpha = \sup \mathscr{O}(M_1)$ and $\alpha \notin \mathscr{O}(M_1)$.

Consequently, we may define for every non-zero limit ordinal α a class $F_\alpha^0 \subset \mathscr{FS}_0$ such that

$$M \in F_\alpha^0 \quad \text{iff} \quad \alpha = \sup \mathscr{O}(M) \text{ and } \alpha \notin \mathscr{O}(M).$$

Now using Corollary 3.2 and Corollary 3.1, we infer that the ordering $\to$ is defined in the class $\overline{\mathscr{FS}} = \{[M]\colon M \in \mathscr{FS}\}$ by the following properties:

(i) for any ordinals $\alpha, \beta \geqslant 1$, $F_\alpha \to F_\beta$ iff $\alpha \leqslant \beta$;

(ii) for any limit ordinal α, F_α^0 is the predecessor of the class F_α;

(iii) CEL is the greatest element in $\overline{\mathscr{FS}}$.

Hence the class $\overline{\mathscr{FS}}$ is well ordered by $\to$ and has a greatest element; thus it is a complete lattice with respect to $\to$. The least element in $\overline{\mathscr{FS}}$ is the class F_1 of all non-empty systems with an empty transition function.

Infinite systems. A non-empty system $M = \langle X, \pi\rangle$ will be called *infinite* iff $D(f_M) = \emptyset$ (or, equivalently, $R(\pi) \subset D(\pi)$) and M has no cyclic computations. Observe that M may have a finite set of states when the transition function is empty. Let $\mathscr{IS}$ be the class of all infinite systems.

Let $M = \langle X, \pi\rangle$ be an arbitrary iterative system. A sequence $(\ldots, x_{-1}, x_0, x_1, \ldots)$ infinite on both sides is said to be a *computation infinite on both sides of M* iff for any integer $i \in Z$, the sequence $(x_i, x_{i+1}, \ldots)$ is a computation of M. In other words, for every $i \in Z$, $\pi(x_i) = x_{i+1}$. We shall write c.i.b. for computations infinite on both sides.

THEOREM 3.7. *If a system $M_1 = \langle X_1, \pi_1\rangle$ has a* c.i.b.*, then for any system $M = \langle X, \pi\rangle \in \mathscr{IS}$ there exists a homomorphism of M into M_1.*

Proof. Let $(\ldots, y_{-1}, y_0, y_1, \ldots)$ be a c.i.b. in M_1. Assume $M \in \mathscr{IS}$ and let S be a m.c.-subsystem of M. Define a relation ϱ_S in S so that $(a, b) \in \varrho_S$ iff there exists a $k \geqslant 0$ such that $\pi^k(a) = \pi^k(b)$. It may be proved that ϱ_S is a congruence in the subsystem $\langle S, \pi\rangle$. Let $x_0 \in S$ be an arbitrary state. For any state $[x]_{\varrho_S}$ in the quotient system S/ϱ_S there exists an integer $m \geqslant 0$ such that either $[\pi^m(x)] = [x_0]$ or $[\pi^m(x_0)] = [x]$.

Define a mapping h_S: $S/\varrho_S \to X_1$ so that

$$h_S([x]) = \begin{cases} y_k & \text{if } [\pi^k(x_0)] = [x] \text{ for } k \geqslant 0, \\ y_{-k} & \text{if } [\pi^k(x)] = [x_0] \text{ for } k > 0. \end{cases}$$

h_S is a homomorphism of $\langle S/\varrho_S, \pi^*\rangle$ into M_1. Thus the mapping $g_S = h_S \cdot \varepsilon_{\varrho_S}$, where ε_{ϱ_S}: $S \to S/\varrho_S$ is the natural homomorphism, is a homomorphism of S into M_1. By Lemma I.3.3, there exists a homomorphism of M into M_1. ■

THEOREM 3.8. *Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be infinite iterative systems. If M has a* c.i.b. *then there exists a homomorphism of M into M_1 iff M_1 has a* c.i.b.

Proof. This proof follows exactly the lines of the proof of Theorem 3.4. It should be remarked that if $(\ldots, x_{-1}, x_0, x_1, \ldots)$ is a c.i.b. in M, then any homomorphism of M into M_1 is 1-1 on the elements of this sequence, since M_1 has no cycles. ■

THEOREM 3.9. *A system $M_1 \in \mathscr{IS}$ has a* c.i.b. *if and only if for any system $M \in \mathscr{IS}$ there exists a homomorphism of M into M_1.*

Proof. The proof follows from Theorems 3.7 and 3.8. ■

It can be seen that infinite systems with c.i.b. have the same properties in $\mathscr{IS}$ as finite systems with c.e.l. in $\mathscr{FS}$. Thus, denoting by CIB the class of all infinite systems with c.i.b., we have

COROLLARY 3.3. CIB *is the greatest element in the family of all complexity classes of infinite systems.* ■

We shall now deal with infinite systems which have no c.i.b. Let the class of such systems be denoted by $\mathscr{IS}_0$.

By an argument similar to the proof of Lemma 3.2 we prove

LEMMA 3.5. *Let* $M = \langle X, \pi\rangle \in \mathscr{IS}_0$. *Then* $X_0 = X - R(\pi)$ *is the least generating set for* M. ■

Now let $M = \langle X, \pi\rangle$ be any (not necessarily connected) infinite system without c.i.b. For every $x \in X$, let $o_M(x)$ be an ordinal such that

(i) if $x \in X - D(\pi)$, then $o_M(x) = 1$,

(ii) if $x \in D(\pi)$, then $o_M(x)$ is the least ordinal greater than any ordinal in the set $\{o_M(y)\colon (\exists k > 0)(\pi^k(y) = x)\}$. In particular, we have $o_M(x) = 0$ for all $x \in D(\pi) - R(\pi)$.

Since $X - R(\pi) = (X - D(\pi)) \cup (D(\pi) - R(\pi))$ (because $R(\pi) \subset D(\pi)$), then, by Lemma 3.5, o_M is well defined on the whole set X. Observe also that for systems with an empty transition function—which are both finite and infinite—the two definitions of the function o_M coincide. Moreover, if $\alpha \in o_M(D(\pi))$, then for any $0 \leqslant \beta \leqslant \alpha$, $\beta \in o_M(D(\pi))$. Thus if $0 \notin o_M(X)$, then $D(\pi) = \emptyset$ and $o_M(X) = \{1\}$.

Using the function o_M, we may define a function O_M which to every computation in M assigns a sequence of ordinals

$$O_M(\bar{x}_0) = (o_M(x_0), o_M(x_1), \ldots), \quad \text{where } \bar{x}_0 = (x_0, x_1, \ldots).$$

Let $O(M)$ be the image of the set C_M of all computations of M. From the definition of o_M we infer that each sequence in $O(M)$ is increasing.

If $\Gamma = (\Gamma_n)_{n\in N}$ and $\Delta = (\Delta_n)_{n\in N}$ are sequences of ordinals, set $\Gamma \leqslant^+ \Delta$ iff $\Gamma_n \leqslant \Delta_n$ for all $n \in N$. This defines an ordering in any family of sequences of ordinals.

If $\Gamma = (\Gamma_n)_{n\in N}$ is any sequence of ordinals, let $\Gamma^{(i)}$ for $i \geqslant 0$ be the sequence $(\Gamma_i, \Gamma_{i+1}, \ldots)$, called a *segment* (*proper*, when $i > 0$) of Γ; if δ is an ordinal, let (δ, Γ) be a sequence such that $(\delta, \Gamma)_0 = \delta$ and $(\delta, \Gamma)^{(1)} = \Gamma$.

The following theorem characterizes those sets of increasing sequences of ordinals which correspond to some infinite iterative system.

THEOREM 3.10. *Let* A *be any set of increasing sequences of ordinals such that*:

(i) *if* $\Gamma \in A$, *then for any* $i \in N$, $\Gamma^{(i)} \in A$;

(ii) *let* $\Gamma \in A$:

(a) *if* $\Gamma_0 = \delta + 1$ *for some ordinal* δ, *then* $(\delta, \Gamma) \in A$;

(b) *if* Γ_0 *is a non-zero limit ordinal, then there exists a sequence of ordinals* $(\delta_\xi)_{\xi<\alpha}$ *such that* $\lim\limits_{\xi<\alpha} \delta_\xi = \Gamma_0$ *and for every* $\xi < \alpha$, $(\delta_\xi, \Gamma) \in A$.

Then there exists a system $M \in \mathscr{IS}_0$ *such that* $A = O(M)$.

Before proving the theorem observe that under the above assumptions the set A has the following property:

LEMMA 3.6. *A sequence* $\Gamma \in A$ *is not a proper segment of any sequence in* A *if and only if* $\Gamma_0 = 0$. ■

Now return to the proof of the theorem.

Proof. Define $M = \langle A, \pi\rangle$, so that $D(\pi) = A$ and $\pi(\Gamma) = \Gamma^{(1)}$ for any $\Gamma \in A$. By (i), π is indeed an operation in A. Clearly every computation in M is infinite and, by the monotonicity of every sequence in A, no computation in M is cyclic. Thus $M \in \mathscr{IS}$. On the other hand, if $(\ldots, \Gamma^{-1}, \Gamma^0, \Gamma^1, \ldots)$ were a c.i.b. in M, then, because of $\Gamma^i = \Gamma^{i-1(1)}$ for all $i \in Z$ and by the monotonicity of all sequences, $\{\Gamma_0^i\}_{i\in Z}$ would be a set of ordinals without a least element. Thus $M \in \mathscr{IS}_0$. We claim that $O(M) = A$.

Observe that for each $\Gamma \in A$, $o_M(\Gamma) = \Gamma_0$. Indeed, if $\Gamma \notin R(\pi)$, then $o_M(\Gamma) = 0$ by the definition of o_M. But $\Gamma \notin R(\pi)$ is equivalent to Γ not being a proper segment of any sequence in A and thus, by Lemma 3.6, $\Gamma_0 = 0$. Therefore $o_M(\Gamma) = \Gamma_0$. On the other hand, assume $\Gamma \in R(\pi)$ and for every $\Sigma \in U(\Gamma)$ (see (6)), $\Sigma \neq \Gamma$, the condition $o_M(\Sigma) = \Sigma_0$ holds.

Suppose $\Gamma_0 = \delta + 1$. It is easy to prove using (ii) and the monotonicity of sequences in A that $\delta + 1$ is the least ordinal greater than any ordinal in the set $\{o_M(\Sigma)\colon (\exists k > 0)(\pi^k(\Sigma) = \Gamma)\} = \{\Sigma_0\colon (\exists k > 0)(\Sigma^{(k)} = \Gamma)\}$.

Similarly, if Γ_0 is a limit ordinal, then by (ii) there exists a sequence $(\delta_\xi)_{\xi<\alpha}$ such that $\lim\limits_{\xi<\alpha} \delta_\xi = \Gamma_0$ and for each $\xi < \alpha$, $\Delta_\xi = (\delta_\xi, \Gamma) \in A$. Then $\Delta_\xi \in U(\Gamma)$, $\Delta_\xi \neq \Gamma$ and $o_M(\Delta_\xi) = \delta_\xi$. Thus, by definition of o_M, $o_M(\Gamma) \geqslant \lim\limits_{\xi<\alpha} \delta_\xi = \Gamma_0$. Besides, for any $\Delta \in U(\Gamma)$, $\Delta \neq \Gamma$, we have $o_M(\Delta) = \Delta_0 < \Gamma_0$ by monotonicity of Δ (since $\Delta^{(k)} = \Gamma$ for some $k > 0$). Hence again Γ_0 is the least ordinal greater than any ordinal in the set $\{o_M(\Sigma)\colon (\exists k > 0)(\pi^k(\Sigma) = \Gamma)\}$, and therefore $o_M(\Gamma) = \Gamma_0$. By Lemma 3.5, this proves that $o_M(\Gamma) = \Gamma_0$ holds for any $\Gamma \in A$.

Suppose $\gamma = (\Gamma, \Gamma^{(1)}, \Gamma^{(2)}, \ldots)$ is a computation in M. Clearly, $O_M(\gamma) = (o_M(\Gamma), o_M(\Gamma^{(1)}), o_M(\Gamma^{(2)}), \ldots) = (\Gamma_0, \Gamma_1, \Gamma_2, \ldots) = \Gamma$. Thus $O(M) \subset A$. Conversely, if $\Gamma \in A$, then $\Gamma = O_M(\gamma)$ for a computation γ defined above. Hence $A \subset O(M)$, and consequently, $A = O(M)$, which proves the theorem. ■

It may easily be checked that for any system $M \in \mathscr{IS}_0$ the set $O(M)$ is a set of increasing sequences of ordinals for which conditions (i) and (ii) of Theorem 3.10 hold. Sets which satisfy the assumptions of that theorem will be called *defining sets*.

The following two lemmas are non-essential modifications of results by M. Novotny ([17], [18]).

LEMMA 3.7. *If* h *is a homomorphism of* $M = \langle X, \pi\rangle$ *into* $M_1 = \langle X_1, \pi_1\rangle$, $M, M_1 \in \mathscr{IS}_0$, *then for any* $x \in D(\pi)$, $o_M(x) \leqslant o_{M_1}(h(x))$. ■

LEMMA 3.8. *Let* $M = \langle X, \pi\rangle$ *and* $M_1 = \langle X_1, \pi_1\rangle$ *be two connected systems,* $M, M_1 \in \mathscr{IS}_0$. *If there exist states* $x_0 \in D(\pi)$ *and* $y_0 \in D(\pi_1)$ *such that for any* $m \geqslant 0$ *the inequality*

$$o_M(\pi^m(x_0)) \leqslant o_{M_1}(\pi_1^m(y_0))$$

holds, then there exists a homomorphism of M *into* M_1. ■

Now let A and B be two defining sets. We shall say that A *precedes* B ($A \prec B$) iff for any $\Gamma \in A$ there exists a $\Delta \in B$ such that $\Gamma \leqslant^{+} \Delta$. This is a reflexive and transitive relation.

Using Lemmas 3.7 and 3.8, we shall prove the following theorem.

THEOREM 3.11. *Let $M = \langle X, \pi\rangle$, $M_1 = \langle X_1, \pi_1\rangle$ and $M, M_1 \in \mathscr{FS}_0$. There exists a homomorphism of M into M_1 if and only if $O(M) \prec O(M_1)$.*

Proof. The theorem is trivial when $D(\pi) = \emptyset$. Assume therefore that $D(\pi) \neq \emptyset$ and let h be a homomorphism of M into M_1. Suppose $\Gamma \in O(M)$. Thus there is a computation $\bar{x}_0 = (x_0, x_1, \ldots)$ in M such that $O_M(\bar{x}_0) = \Gamma$. By Theorem I.3.1 the sequence $\overline{h(x_0)} = (h(x_0), h(x_1), \ldots)$ is a computation of M_1: let $\Delta = O_{M_1}(\overline{h(x_0)})$. Then, by Lemma 3.7, $\Gamma \leqslant^{+} \Delta$. Γ being an arbitrary element of $O(M)$, this proves that $O(M) \prec O(M_1)$.

Conversely, let S be an m.c.-subsystem of M. If $\operatorname{card}(S) = 1$ (i.e. S consists of one isolated state only), let h_S be any mapping of S into X_1. If $\operatorname{card}(S) > 1$, then assume that $\bar{x}_0 = (x_0, x_1, \ldots)$ is a computation in S and let $\Gamma = (\Gamma_n)_{n\in N} = O_M(\bar{x}_0)$. By assumption there exists a sequence $\Delta = (\Delta_n)_{n\in N} \in O(M_1)$ such that $\Gamma \leqslant^{+} \Delta$. Let $\bar{y}_0 = (y_0, y_1, \ldots)$ be any computation with $O_M(\bar{y}_0) = \Delta$. Then from Lemma 3.8 it follows that there exists a homomorphism h_S of S into M_1, since for every $n \in N$,

$$o_M(\pi^n(x_0)) = \Gamma_n \leqslant \Delta_n = o_{M_1}(\pi_1^n(y_0)).$$

Hence, for any m.c.-subsystem S in M there exists a homomorphism h_S of S into M_1, which by Lemma I.3.3 ensures the existence of a homomorphism of M into M_1. ■

By Theorem 3.11 we infer that for any $M, M_1 \in \mathscr{FS}_0$,

$$M \leftrightarrow M_1 \quad \text{iff} \quad O(M) \prec O(M_1) \,\&\, O(M_1) \prec O(M).$$

Thus any defining set A defines indeed a complexity class of infinite systems without c.i.b. (which will be denoted by $|A|$) such that

$$M \in |A| \quad \text{iff} \quad O(M) \prec A \,\&\, A \prec O(M).$$

Moreover,

$$|A| \to |B| \quad \text{iff} \quad A \prec B$$

and consequently,

$$|A| = |B| \quad \text{iff} \quad A \prec B \,\&\, B \prec A.$$

Thus $[M] = |O(M)|$.

If $\{A_i\}_{i\in I}$ is any family of defining sets, then $\bigcup_{i\in I} A_i$ is a defining set again and $\sup\{|A_i|\}_{i\in I}$—which exists by Theorem 3.2—is the class $\left|\bigcup_{i\in I} A_i\right|$. Observe also that $|\emptyset|$ ($\emptyset = O(M)$ for a system M with an empty transition function) is the least element in the family $\overline{\mathscr{FS}}$ of complexity classes of infinite systems. Therefore $\overline{\mathscr{FS}}$ is a complete lattice under $\to$, the greatest element being CIB.

Cyclic systems. Let $M = \langle X, \pi\rangle$ be a non-empty system. Let d be a partial function of X into N such that for $x \in X$

$$d(x) = \begin{cases} \min\{k\colon \pi^k(x) = x \,\&\, k \geqslant 1\} & \text{if this is defined,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

A non-empty system $M = \langle X, \pi\rangle$ will be called *cyclic* iff for every m.c.-subsystem S of M the set $d(S)$ is non-empty. Let the class of all cyclic systems be denoted by $\mathscr{CS}$.

LEMMA 3.9. *Let $M = \langle X, \pi\rangle$, $M_1 = \langle X_1, \pi_1\rangle$ and $M, M_1 \in \mathscr{CS}$. Then there exists a homomorphism of M into M_1 if and only if for any $m \in d(X)$ there exists an $m_1 \in d(X_1)$ such that m_1 divides m.*

Proof. Assume that h is a homomorphism of M into M_1 and let $m \in d(X)$. Thus there is a state $x \in X$ with $d(x) = m$, i.e. $\pi^m(x) = x$. Then

$$h(x) = h(\pi^m(x)) = \pi_1^m(h(x)).$$

Hence $m \geqslant d(h(x))$. It is easily proved that $d(h(x))$ must divide m.

Conversely, assume that S is an m.c.-subsystem of M and let $d(S) = \{m\}$ (observe that the function d is a partial constant function on each m.c.-subsystem). Suppose S_1 is an m.c.-subsystem of M_1 such that $d(S_1) = \{m_1\}$ and m_1 divides m. Take any state $x_S \in S$ with $d(x_S) = m$ and a state $y_{S_1} \in S_1$ with $d(y_{S_1}) = m_1$ and define a mapping $h_S\colon S \to S_1$ as follows:

$$h_S(x) = \pi_1^{m-k_x}(y_{S_1}),$$

where k_x is an integer such that $k_x \equiv k \pmod{m}$, $0 \leqslant k_x < m$ and $\pi^k(x) \doteq x_S$ (k_x is uniquely determined for each $x \in S$). It may be verified that h_S is a well-defined homomorphism of S into M_1. By Lemma I.3.3, there exists a homomorphism of M into M_1. ■

For any subset A of the set N^+ of all positive integers we shall denote by $A_{\min}$ the set of all minimal elements in A with the divisibility ordering (m precedes n iff m divides n) in N^+. Clearly, $A_{\min} \neq \emptyset$ when $A \neq \emptyset$. A is an antichain when $A = A_{\min}$.

THEOREM 3.12. *For any cyclic systems $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$, $M \leftrightarrow M_1$ if and only if $d(X)_{\min} = d(X_1)_{\min}$.*

Proof. It is easily proved that if $d(X)_{\min} = d(X_1)_{\min}$, then for any $m \in d(X)$ there exists an $m_1 \in d(X_1)$ such that m_1 divides m and conversely: for any $m_1 \in d(X_1)$ there exists an $m \in d(X)$ which divides m_1. Thus there exist homomorphisms of M into M_1 and of M_1 into M, by Lemma 3.9.

On the other hand, assume that there exists a homomorphism of M into M_1 and let $m \in d(X)_{\min} \subset d(X)$. By Lemma 3.9, there exists an $m_1 \in d(X_1)$ which divides m. Because of the properties of the divisibility ordering in N^+ we may assume that m_1 is minimal in $d(X_1)$, i.e. $m_1 \in d(X_1)_{\min}$. Moreover, if there exists a homomorphism of M_1 into M, then there exists an element $m_2 \in d(X)$ which divides m_1. Hence m_2 divides m which, by the minimality of m, implies $m_2 = m = m_1$. Thus $m \in d(X_1)_{\min}$

and $d(X)_{\min} \subset d(X_1)_{\min}$. Similarly we prove $d(X_1)_{\min} \subset d(X)_{\min}$ and in consequence these two sets are equal. ■

Theorem 3.12 implies that every complexity class of cyclic systems is uniquely determined by an antichain in N^+ with the divisibility ordering. The class determined by an antichain A will be denoted by $|A|$ and the antichain which defines the class $[M]$ for a cyclic system M will be denoted by $A_{[M]}$. Observe that for every antichain A there exists a cyclic system $M = \langle X, \pi\rangle$ such that $A = d(X)_{\min}$. Thus complexity classes of cyclic systems are in a 1-1 correspondence with antichains in N^+.

It should be remarked that the family $\overline{\mathscr{CS}}$ of all complexity classes of cyclic systems is not a complete lattice: take the family of one-element antichains $\{p_n\}_{n\in N}$, where p_n is the nth prime number; then there is no lower bound for the family of complexity classes $|\{p_n\}|$, since there is no natural number which could be divided by all prime numbers. However, $\rightarrow$ is a lattice ordering in $\overline{\mathscr{CS}}$.

THEOREM 3.13. *The relation $\rightarrow$ is a lattice ordering in $\overline{\mathscr{CS}}$ and*

$$A_{[M]\vee[M_1]} = (A_{[M]} \cup A_{[M_1]})_{\min},$$
$$A_{[M]\wedge[M_1]} = \{\text{l.c.m.}(m, m_1)\colon m \in A_{[M]},\ m_1 \in A_{[M_1]}\}_{\min},$$

where $\vee$, $\wedge$ are the lattice operations and l.c.m. (m, m_1) *is the least common multiple of m and m_1.*

Proof. Clearly the class $|(A_{[M]}\cup A_{[M_1]})_{\min}|$ is an upper bound for $\{[M], [M_1]\}$. If C is an antichain such that $|C|$ is also an upper bound for that set, then for any $m \in A_{[M]}\cup A_{[M_1]}$ (and in particular for all $m \in (A_{[M]}\cup A_{[M_1]})_{\min}$) there exists an $m_1 \in C$ which divides m. This means exactly (by Lemma 3.9) that $|(A_{[M]}\cup A_{[M_1]})_{\min}| \rightarrow |C|$, which proves that the former class is the least upper bound for $\{[M], [M_1]\}$.

Now let $D(M, M_1) = \{\text{l.c.m.}(m, m_1)\colon m \in A_{[M]},\ m_1 \in A_{[M_1]}\}_{\min}$. If $m_2 \in D(M, M_1)$, then there exist an $m \in A_{[M]}$ and an $m_1 \in A_{[M_1]}$ such that $m_2 = \text{l.c.m.}(m, m_1)$. Therefore m and m_1 divide m_2, which proves that $D(M, M_1)$ is a lower bound for $\{[M], [M_1]\}$. If B is an antichain and $|B|$ is also a lower bound, then for any $b \in B$ there exist an $m \in A_{[M]}$ and an $m_1 \in A_{[M_1]}$ which divide b. Therefore also l.c.m.(m, m_1) divides b, which implies $|B| \rightarrow |D(M, M_1)|$ and $D(M, M_1)$ is the antichain of the complexity class $[M]\wedge[M_1]$. ■

By Theorem 3.2, $\overline{\mathscr{CS}}$ is also a complete upper semilattice. The greatest element in $\overline{\mathscr{CS}}$ is the class $|\{1\}|$, i.e. the complexity class of all cyclic systems which have one-element cycles.

Now let $M = \langle X, \pi\rangle$ be an arbitrary non-empty system. Let $X^{(1)}$ be the subset of all states of finite computations in M (i.e. $X^{(1)} = D(f_M)\cup R(f_M)$), $X^{(2)}$—the subset of all states of infinite and non-cyclic computations, and $X^{(3)}$—the subset of all states which appear in cyclic computations of M. Clearly, if $i \neq j$, then $X^{(i)}\cap X^{(j)} = \emptyset$ and $X^{(1)}\cup X^{(2)}\cup X^{(3)} = D(\pi)\cup R(\pi)$.

For $i = 1, 2, 3$ let $M^{(i)}$ be the subsystem of M with the set of states $X^{(i)}$, i.e. $M^{(i)} = \langle X^{(i)}, \pi\rangle$.

THEOREM 3.14. *Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$, where $X_1 \neq \emptyset$, be arbitrary systems such that $X^{(3)} = X_1^{(3)} = \emptyset$. Then there exists a homomorphism of M into M_1 if and only if there exist homomorphisms of $M^{(1)}$ into M_1 and of $M^{(2)}$ into M_1.*

Proof. The necessity of the condition follows from Lemma I.3.2. To prove sufficiency, assume that h_1 is a homomorphism of $M^{(1)}$ into M_1 and h_2 is a homomorphism of $M^{(2)}$ into M_1. Let g be any mapping of $X - (D(\pi)\cup R(\pi))$ into X_1 (which always exists, since $X_1 \neq \emptyset$). Then $h = h_1\cup h_2\cup g$ is a well-defined homomorphism of M into M_1. ■

THEOREM 3.15. *Let $M = \langle X, \pi\rangle$ and $M_1 = \langle X_1, \pi_1\rangle$ be systems such that $X_1^{(3)} \neq \emptyset$. Then a homomorphism of M into M_1 exists if and only if there exist homomorphisms of $M^{(1)}$ into M_1 and of $M^{(3)}$ into M_1.*

If $X^{(3)} \neq \emptyset$ and $X_1^{(3)} = \emptyset$, then there is no homomorphism of M into M_1.

Proof. Assume $X_1^{(3)} \neq \emptyset$. Then the necessity of the condition follows again from Lemma I.3.2. On the other hand, assume that there exist homomorphisms h_1 of $M^{(1)}$ into M_1 and h_3 of $M^{(3)}$ into M_1. We claim that there exists also a homomorphism h_2 of $M^{(2)}$ into M_1.

If $X^{(2)} = \emptyset$, then the empty mapping is a homomorphism of $M^{(2)}$ into M_1; assume therefore that $X^{(2)} \neq \emptyset$. Since $X_1^{(3)} \neq \emptyset$, there exists a state $x_1 \in X_1$ such that $x_1 \in D(d)$ and $d(x_1) = m$ for some $m > 0$.

Let S be any m.c.-subsystem of $M^{(2)}$ (by the definition of $M^{(2)}$, $S \subset D(\pi)$) and x_0 an arbitrary state in S. Define a mapping h_S of S into X_1 as follows:

(i) let $h_S(x_0) = x_1$;

(ii) if for some $x \in S$ we have already defined $h_S(x) = \pi_1^k(x_1)$, then let $h_S(\pi(x)) = \pi_1^{k+1}(x_1)$ and $h_S(x') = \pi_1^{m+k-1}(x_1)$ if $\pi(x') = x$.

h_S is a well-defined mapping on S and it is also a homomorphism. Therefore by Lemma I.3.3 there exists a homomorphism h_2 of $M^{(2)}$ into M_1. Then if g is any mapping of $X - (D(\pi)\cup R(\pi))$ into X_1, the function $h = h_1\cup h_2\cup h_3\cup g$ is a homomorphism of M into M_1.

Assume now $X^{(3)} \neq \emptyset$. Thus there exists a cyclic computation in M which by Theorem I.3.1 is transformed by any homomorphism into a cyclic computation of M_1. Thus, if such a homomorphism exists, then $X_1^{(3)} \neq \emptyset$. ■

Let Emp denote the class of all non-empty systems with an empty transition function (this is the class F_1 if we regard such systems as finite and it is the class $|\emptyset|$ if we regard them as infinite systems). For any system M_1 and $M \in$ Emp there exists a homomorphism of M into M_1. Thus Emp is the least element in the family of all complexity classes.

Assume that M is a non-empty system and $M \notin$ Emp. By Theorems 3.14 and 3.15, the complexity class of M may be univocally determined by a pair (A, B) such that

$$A = \begin{cases} [M^{(1)}] & \text{if } X^{(1)} \neq \emptyset, \\ \emptyset & \text{if } X^{(1)} = \emptyset, \end{cases}$$

and

$$B = \begin{cases} [M^{(3)}] & \text{if } X^{(3)} \neq \emptyset, \\ [M^{(2)}] & \text{if } X^{(3)} = \emptyset \neq X^{(2)}, \\ \emptyset & \text{if } X^{(3)} = \emptyset = X^{(2)}, \end{cases}$$

where at least one element of the pair is not $\emptyset$. Then, if we take $\emptyset \to [M]$ for any system M and $\emptyset \to \emptyset$, the ordering $\to$ is, again by Theorems 3.14 and 3.15, defined on the components, i.e.

$$(A, B) \to (C, D) \quad \text{iff} \quad A \to C \text{ and } B \to D.$$

Observe here that by Theorem 3.15 every complexity class of infinite systems precedes any complexity class of cyclic systems. Thus we obtain an ordered classification of systems which by Theorem 3.2 is a complete upper semilattice—the greatest element being the class (CEL, $|\{1\}|$)—with a least element, which is the class Emp. Thus it is a complete lattice.

Returning to programs in SPC, observe that the relations of weak similarity and similarity are equivalences. For any program φ in an SPC M the class of programs weakly similar to φ is sufficiently described by $[M(\varphi)]$, while the class of programs similar to φ is determined by a pair $[M(\varphi)]$, $[M^*(\varphi)]$.

4. Simplification of programs in SPC

One of the methods of program optimization consists in decomposing programs into subprograms and performing minimization based on such a decomposition. This occurs e.g. when large open subprograms are replaced by closed subprograms. Such a situation will be considered here. We shall describe an optimizing procedure and prove its correctness. The considerations presented here are simplification of notions as well as a generalization of results given in [32].

By M we shall denote a stored program computer fixed for our considerations. If $p = \langle \varphi, a \rangle$ is an l-program, then the set

$$\{c \in C^p\colon (\exists m)_N(\pi^m(c)(l) \notin D(\varphi))\}$$

will be denoted by B^p. If X is a set of l-programs, then the address of X is the set of all addresses of p, where $p \in X$([4]). The basic notion for further considerations is the notion of an F-program.

An *F-program* is any l-program $p = \langle \varphi, a \rangle$ such that

(i) $(\forall c)_{C^p - B^p}(\forall t)_N(t < n_c + 1 \Rightarrow \pi^t(c) \in c^\varphi)$, where $n_c = (\mu k)(\pi^k(c) \notin D(\pi))$;([5])

(ii) $(\forall c)_{B^p}(\forall t)_N\big(t \leqslant (\mu p)(\pi^p(c)(l) \notin D(\varphi)) \Rightarrow \pi^t(c) \in C^\varphi\big)$.

The meaning of an F-program is described by a partial function which is called a reduced realization of an F-program.

([5]) i.e. n_c is the smalest number k such that $\pi^k(c) \notin D(\pi)$.

([4]) the address of $p = \langle \varphi, a \rangle$ is a.

A reduced realization of an F-program $p = \langle \varphi, a \rangle$, denoted by δ^p, is a partial function such that

(i) $D(\delta^p) = C^p \cap D(f_M)$;

(ii) if $c \in D(\delta^p)$, then

$$\delta^p(c) = \begin{cases} f_M(c) & \text{if for } k \leqslant n_c,\ \pi^k(c)(l) \in D(\varphi), \\ \pi^m(c) & \text{otherwise,} \\ & \text{where } m = \mu p\big(\pi^p(c)(l) \notin D(\varphi)\big). \end{cases}$$

A subprogram of an F-program $p = \langle \varphi, a \rangle$ is any F-program $p' = \langle \psi, b \rangle$ such that

(i) $D(\psi) \subset D(\varphi)$;

(ii) if $x \in D(\psi) - \{b\}$, then there is no $c \in C^p$ such that for some $k \in N$

$$\pi^k(c)(l) \in D(\varphi) - D(\psi), \quad \pi^{k+1}(c)(l) = x$$

and

$$\pi(c) \in C^\varphi, \ \ldots, \ \pi^{k+1}(c) \in C^\varphi.$$

If $p = \langle \varphi, a \rangle$ is an F-program, we shall call a *partition* of p an arbitrary se $\{\langle \varphi_1, a_1 \rangle, \ldots, \langle \varphi_k, a_k \rangle\}$ of subprograms of p such that

$$\bigcup_{i=1}^{k} D(\varphi_i) = D(\varphi);$$

$$D(\varphi_i) \cap D(\varphi_j) = \emptyset \quad \text{for } i \neq j \ (i, j = 1, \ldots, k).$$

Let $P = \{p_0, \ldots, p_k\}$ be a partition of an F-program $p = \langle \varphi, a \rangle$. A partial function Φ^p such that

$$\Phi^p(p_i, c) = \begin{cases} p_k & \text{if } (c \in C^{p_i}) \wedge \big(\delta^{p_i}(c)(l) = \text{address}(p_k)\big) \wedge (i \neq k), \\ \text{undefined} & \text{otherwise,} \end{cases}$$

will be called the *mahrosuccessor function* in the partition P of p.

We shall compare F-programs using the ϱ-equivalence relation. Let $p = \langle \varphi, a \rangle$ $q = \langle \psi, b \rangle$ be F-programs and $\varrho \subset C \times C$ (where C is the memory of the SPC under consideration) an equivalence relation. The ϱ-equivalence relation, denoted by $\underset{\varrho}{\sim}$, is defined as follows:

$$p \underset{\varrho}{\sim} q \text{ iff } (\forall c)_{C^p}(\forall c')_{C^q}\big[(c \varrho c' \Rightarrow \delta^p(c) \varrho \delta^q(c')) \wedge \big(\delta^p(c)(l) \in D(\varphi) \equiv \delta^q(c')(l) \in D(\psi)\big)\big] \wedge D\big((C^p \times C^q) \cap \varrho\big) = C^p \wedge R\big((C^p \times C^q) \cap \varrho\big) = C^q.$$

We say that an F-program q is *ϱ-regular* iff for any $c, c' \in C^q$ such that $c \underset{\varrho}{\sim} c'$ we have $\delta^q(c)(l) = \delta^q(c')(l)$.

Let $\langle \varphi, a \rangle$, $\langle \psi, b \rangle$ be F-programs and P, P' their partitions and let $\underset{\varrho}{\sim}$ be the ϱ-equivalence relation of F-programs. A relation $\mu \subset P \times P'$ is a *ϱ-comorphism* in $P \times P'$ if for any $\langle p, q \rangle \in P \times P'$

(i) $p \mu q \Rightarrow p \underset{\varrho}{\sim} q$;

(ii) $p \mu q \Rightarrow (\forall c)_{C^p}(\forall c')_{C^q}[c \varrho c' \Rightarrow \Phi^P(p, c) \mu \Phi^{P'}(q, c')]$;

(iii) $D(\mu) = P$.

If there exists a ϱ-comorphism μ in $P\times P'$ we shall write $P \underset{\varrho}{\approx} P'$.

LEMMA 4.1. *Let P, P' be partitions of F-programs $\langle\varphi, a\rangle$ and $\langle\psi, b\rangle$, respectively, and let $\underset{\varrho}{\sim}$ be the ϱ-equivalence relation of F-programs. Let*

(i) $\mu_0 = \{\langle p, q\rangle \in P\times P' : p \underset{\varrho}{\sim} q\}$;

(ii) $\mu_i = \mu_{i-1} \cap \{\langle p, q\rangle \in P\times P' : (\forall c)_{C^p}(\forall c')_{C^q}[c\varrho c' \Leftrightarrow \langle \Phi^P(p, c), \Phi^{P'}(q, c')\rangle \in \mu_{i-1}]\}$ *for* $i \geqslant 1$;

(iii) $\mu_\infty = \bigcap_{i\geqslant 0} \mu_i$.

Then (a) $(\exists k)_N(\mu_\infty = \mu_k)$,

(b) *μ_∞ fulfils conditions* (i) *and* (ii) *of the definition of a ϱ-comorphism,*

(c) *if μ_∞ does not fulfil condition* (iii) *of the definition of ϱ-comorphism, then there is no ϱ-comorphism in $P\times P'$.*

Proof. Since the set $P\times P'$ is finite, condition (a) holds. From the definition of μ_∞ we have: $\mu_\infty \subset \mu_0$. Hence μ_∞ has property (i) from the definition of ϱ-comorphism. Let $p\mu_\infty q$, $c \in C^p$, $c' \in C^q$ and $c\varrho c'$. We have $p\mu_i q$ for $i = 1, 2, \ldots$ and hence by the definition of μ_i

$$\Phi^P(p, c)\mu_j\Phi^{P'}(q, c') \quad \text{for} \quad j = 0, 1, \ldots$$

This means that

$$\Phi^P(p, c)\mu_\infty\Phi^{P'}(q, c').$$

It is easy to prove that if μ is a ϱ-comorphism, then $\mu \subset \mu_\infty$. Suppose that there exists a ϱ-comorphism $\mu \subset P\times P'$ and $D\mu_\infty \neq P$. Then we have, for some $p \in P$ and $q \in P'$,

$$\langle p, q\rangle \in \mu \quad \text{and} \quad \langle p, q\rangle \notin \mu_\infty,$$

and so $\langle p, q\rangle \in \mu - \mu_\infty$, which contradicts the condition $\mu \subset \mu_\infty$. This means that if $D\mu_\infty \neq P$, then a ϱ-comorphism in $P\times P'$ does not exist. ■

In [32] the following problem is considered:

An equivalence relation $\varrho \subset C\times C$ is given. We have to determine a procedure which transforms an F-program $\langle\varphi, a\rangle$ and its partition P into an F-program $\langle\psi, b\rangle$ and its partition Q_0 such that

(I) $P \underset{\varrho}{\approx} Q_0$;

(II) $\operatorname{card}(Q_0) \leqslant \operatorname{card}(Q)$ for any Q such that Q is a partition of an F-program $\langle\psi', b'\rangle$ and $P \underset{\varrho}{\approx} Q$.

The following procedure transforming an F-program $\langle\varphi, a\rangle$ and its partition P into a minimal ϱ-comorphic F-program and its partition is proposed in [32]:

(i) Find μ_∞. (We assume here that $\langle\varphi, a\rangle = \langle\psi, b\rangle$ in the definition of μ_∞.)

(ii) Find P/μ_∞.

(iii) Let $(q_1, \ldots, q_k)$ be a sequence of elements from distinct equivalence classes of μ_∞ such that $k = \operatorname{card}(P/\mu_\infty)$ and $a \in \operatorname{address}[q_1]$. Find ϱ-regular F-programs with pairwise disjoint domains

$$p_1 = \langle\varphi_1, a_1\rangle, \ldots, p_k = \langle\varphi_k, a_k\rangle$$

so that for i, j $(i, j = 1, \ldots, k)$: $p_i \underset{\varrho}{\sim} q_i$ and for any $\hat{c} \in C^{p_i}$, $c \in C^{q_i}$ if $\hat{c}\varrho c$, then

(a) if $\delta^{q_i}(c)(l) \notin D(\varphi)$, then $\delta^{p_i}(c)(l) \notin \bigcup_{i=1}^{k} D(\varphi_i)$;

(b) if $\delta^{q_i}(c)(l) \in \operatorname{address}[q_j]_{\mu_\infty}$, then $\delta^{p_i}(\hat{c})(l) = \operatorname{address}(p_j)$.

THEOREM 4.1. *The F-program $\langle \bigcup_{i=1}^{k} \varphi_i, a_1\rangle$ and its partition*

$$Q_0 = \{\langle\varphi_1, a_1\rangle, \ldots, \langle\varphi_k, a_k\rangle\}$$

defined as the result of the transformation of an F-program $\langle\varphi, a\rangle$ and its partition P by means of the procedure defined as above, have properties (I) *and* (II).

Proof. All the symbols used here have the same meaning as in the above procedure. Let us consider F-programs $p = \langle\varphi, a\rangle$, $q = \langle \bigcup_{i=1}^{k} \varphi_i, a_1\rangle$ and their partitions P and Q_0, respectively. We have $P \underset{\varrho}{\approx} Q_0$. Indeed, let $\mu \subset P\times Q_0$ be the relation defined as follows:

$$\mu = \bigcup_{i=1}^{k} \{\langle p', p_i\rangle : p'\mu_\infty q_i \ \&\ p' \in P\}.$$

From the description of the procedure we conclude that μ is a ϱ-comorphism in $P\times Q_0$.

From the definition of Q_0 we have $\operatorname{card}(Q_0) = \operatorname{card}(P/\mu_\infty)$. Suppose that there exists an F-program $q' = \langle\psi', b'\rangle$ and its partition Q such that $P \underset{\varrho}{\approx} Q$ and $\operatorname{card}(Q) < \operatorname{card}(Q_0)$. Then there exists a ϱ-comorphism μ'' in $P\times Q$. Since $\operatorname{card}(Q) < \operatorname{card}(Q_0)$ and $D\mu'' = P$, there exist $p', p'' \in P$ and $q' \in Q$ such that

$$\sim p'\mu_\infty p; \tag{1}$$

$$p'\mu''q' \wedge p''\mu''q'. \tag{2}$$

From (1) it follows that for some i

$$\langle p', p''\rangle \notin \mu_i, \tag{3}$$

where μ_i is defined as in the Lemma 1.

By (2), we obtain $p' \underset{\varrho}{\sim} q'$ and $p'' \underset{\varrho}{\sim} q'$. Hence $p' \underset{\varrho}{\sim} p''$, and so $\langle p', p''\rangle \in \mu_0$.

Let $c'\varrho c''$ and $c' \in C^{p'}$, $c'' \in C^{p''}$. By the definition of $\underset{\varrho}{\sim}$, it follows that there exist $\hat{c}' \in C^{q'}$ and $\hat{c}'' \in C^{q'}$ such that

$$c'\varrho'\hat{c}' \quad \text{and} \quad c''\varrho\hat{c}''.$$

Hence $\hat{c}'\varrho\hat{c}''$. By the definition of a ϱ-comorphism we have

$$\Phi^P(p', c')\mu''\Phi^Q(q', \hat{c}') \quad \text{and} \quad \Phi^P(p'', c'')\mu''\Phi^Q(q', \hat{c}''). \tag{4}$$

Since q' is a ϱ-regular F-program and $\hat{c}'\varrho\hat{c}''$ as well as $\hat{c}' \in C^{q'}$, $\hat{c}'' \in C^{q'}$, we have

$$\Phi^Q(q', \hat{c}') = \Phi^Q(q', \hat{c}''), \tag{5}$$

i.e., $\Phi^Q(q', \hat{c}')$ is defined iff $\Phi^Q(q', \hat{c}'')$ is defined, and if both $\Phi^Q(q', \hat{c}')$ and $\Phi^Q(q', \hat{c}'')$ are defined, then equality (5) holds. If $\Phi^Q(q', \hat{c}')$ is undefined, then also $\Phi^P(p'', c'')$

and $\Phi^P(p', c')$ are undefined, and so $\langle p', p''\rangle \in \mu_\infty$. This contradicts (1). Hence $\Phi^Q(q', c')$ is defined.

From (4) we obtain

$$\Phi^P(p', c') \underset{\varrho}{\sim} \Phi^Q(q', \hat{c}') \quad \text{and} \quad \Phi^P(p'', c'') \underset{\varrho}{\sim} \Phi^Q(q', \hat{c}'').$$

By (5), we have

$$\Phi^P(p', c') \underset{\varrho}{\sim} \Phi^P(p'', c''),$$

so $\langle p', p''\rangle \in \mu_1$.

In an analogous way we obtain $\langle p', p''\rangle \in \mu_2, \ldots, \langle p', p''\rangle \in \mu_i$, which contradicts (1). It means that condition (II) holds. ■

EXAMPLE. Let $M = \langle C, R, \varrho, \varkappa, \lambda, l\rangle$ be a stored program machine, where the set of addresses is the set N, $l = 0$, $B = N \cup \{*\}$ $(* \notin N)$ and the memory C is the set of all partial functions from A to B with finite domain and such that:

$$\begin{aligned} c(l) &\in D(c) \quad \text{for } c \in C,\\ 100 &\leqslant c(1) \leqslant 199,\\ 200 &\leqslant c(2) \leqslant 299,\\ 300 &\leqslant c(3) \leqslant 499,\\ c(500) &= *. \end{aligned}$$

Let $\lambda(c(l)) = c(l)+1$ for $c \in C$. The alphabet of the language of instructions is the following:

$$\mathscr{A} = \{\text{STOP}, +, \dot{-}, \text{sg}, \overline{\text{sg}}, \cdot, f_1, (,), \alpha, \to\},$$

where STOP, $+$, $\dot{-}$, sg, $\overline{\text{sg}}$, $\cdot$, f_1 are names of operations defined as follows:

(1) $\text{ar}(+) = \text{ar}(-) = \text{ar}(\cdot) = 2$,

(2) $\text{ar}(\text{sg}) = \text{ar}(\overline{\text{sg}}) = \text{ar}(f_1) = \text{ar}(\text{STOP}) = 1$,

(3) $D(\text{STOP}) = \emptyset$,

(4) $$f(b, b') = \begin{cases} b \bar{f} b' & \text{if } b, b' \in N,\\ * & \text{otherwise,}\end{cases}$$

where $f \in \{+, \dot{-}, \cdot\}$ and $\bar{f}$ is the addition, substraction, and multiplication operation in N, respectively,

(5) $$\text{sg}(b) = \begin{cases} 0 & \text{if } b = 0\\ 1 & \text{otherwise,}\end{cases} \qquad \overline{\text{sg}}(b) = \begin{cases} 1 & \text{if } b = 0\\ 0 & \text{otherwise,}\end{cases}$$

(6) $$f_1(b) = \begin{cases} 1 & \text{if } b = *\\ 0 & \text{otherwise.}\end{cases}$$

We assume that the following instructions are in the set R:

(i) $a \to b$,

(ii) $+(\alpha(a), 1) \to a$,

(iii) $\alpha(\alpha(a)) \to \alpha(b)$,

(iv) $\alpha(a) \to \alpha(b)$,

(v) $$+\Big(\cdot\big(a, \text{sg}\big(\dot{-}\big(\alpha(\alpha(j)), \alpha(\alpha(i))\big)\big)\big), \cdot\Big(b, \overline{\text{sg}}\Big(\dot{-}\Big(+\big(\alpha(\alpha(i)), 1\big), \alpha(\alpha(j))\Big)\Big)\Big)\Big) \to l,$$

(vi) $$+\Big(\cdot\big(a, f_1\big(\alpha(\alpha(i))\big)\big), \cdot\big(b, \overline{\text{sg}}\big(f_1\alpha(\alpha(i))\big)\big)\Big) \to l,$$

(vii) $$+\Big(\cdot\big(a, \overline{\text{sg}}\big(\alpha(i)\big)\big), \cdot\big(b, \text{sg}(\alpha(i))\big)\Big) \to l,$$

(viii) $\text{STOP}(\alpha(0)) \to l$,

where $a, b, i, j \in N$.

In the sequel instructions of the form (v)–(vii) are denoted by $\alpha(\alpha(i)) < \alpha(\alpha(j))[a, b]$, $\alpha(i) = *[a, b]$, $\alpha(i) = 0[a, b]$, respectively, and instead of $\text{STOP}(\alpha(0)) \to l$ we write STOP.

Let $X = (x_0, \ldots, x_{m-1})$, $Y = (y_0, \ldots, y_{n-1})$, where $0 < m, n < 100$ and $x_i, y_i \in N$ and $x_i < x_{i+1}$ for $i = 0, 1, \ldots, m-2$, $y_i < y_{i+1}$ for $i = 0, 1, \ldots, n-1$. We construct an l-program which transforms X and Y into a sequence $Z = (z_0, \ldots, z_{m+n-1})$ such that (a) $z_i \leqslant z_{i+1}$ for $i = 0, \ldots, m+n-2$, (b) z_i is a member of X or Y for $i = 0, \ldots, m+n-2$.

Element x_i of X is stored at the address $i+100$, element y_j of Y is stored at the address $j+200$, element z_k of Z is stored at the address $k+300$. The end of each sequence X, Y, Z is marked by $*$. The l-program and its partition are given in Fig. 2. It is easy to observe that this l-program is an F-program. 1 is the address of this F-program.

We shall consider the following equivalence relation: $c \varrho c'$ iff $c|D = c'|D$, where $D = \{1, 2, 3, 100, \ldots, 499\}$. It is easy to observe that

$$\begin{aligned} &p_2 \underset{\varrho}{\sim} p_{12}\\ &p_5 \underset{\varrho}{\sim} p_9\\ &p_3 \underset{\varrho}{\sim} p_4 \underset{\varrho}{\sim} p_6 \underset{\varrho}{\sim} p_7 \underset{\varrho}{\sim} p_{10} \underset{\varrho}{\sim} p_{11} \underset{\varrho}{\sim} p_{13} \underset{\varrho}{\sim} p_{14}. \end{aligned}$$

We have $\mu_0 = \{\langle p, p\rangle : p \in P\} \cup K \cup K'$, where

$$\begin{aligned} K = \{&\langle p_2, p_{12}\rangle, \langle p_5, p_9\rangle,\\ &\langle p_3, p_4\rangle, \langle p_3, p_6\rangle, \langle p_3, p_7\rangle, \langle p_3, p_{10}\rangle, \langle p_3, p_{11}\rangle,\\ &\langle p_3, p_{13}\rangle, \langle p_3, p_{14}\rangle,\\ &\langle p_4, p_6\rangle, \langle p_4, p_7\rangle, \langle p_4, p_{10}\rangle, \langle p_4, p_{11}\rangle, \langle p_4, p_{13}\rangle, \langle p_4, p_{14}\rangle,\\ &\langle p_6, p_7\rangle, \langle p_6, p_{10}\rangle, \langle p_6, p_{11}\rangle, \langle p_6, p_{13}\rangle, \langle p_6, p_{14}\rangle,\\ &\langle p_7, p_{10}\rangle, \langle p_7, p_{11}\rangle, \langle p_7, p_{13}\rangle, \langle p_7, p_{14}\rangle,\\ &\langle p_{10}, p_{11}\rangle, \langle p_{10}, p_{13}\rangle, \langle p_{10}, p_{14}\rangle,\\ &\langle p_{11}, p_{13}\rangle, \langle p_{11}, p_{14}\rangle, \langle p_{13}, p_{14}\rangle\}, \end{aligned}$$

$$K' = \{\langle p, q\rangle : \langle q, p\rangle \in K\},$$

$P = \{p_0, \ldots, p_{14}\}$ is a partition given in Fig. 2.

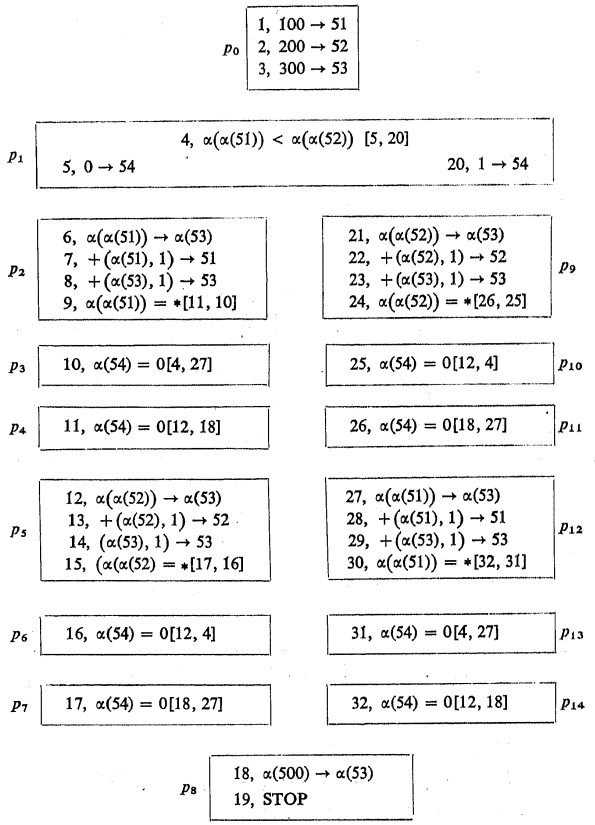


Fig. 2

One can verify that

$$\mu_1 = \{\langle p, p\rangle : p \in P\} \cup K_1 \cup K_1',$$

where

$$K_1 = \{\langle p_2, p_{12}\rangle, \langle p_3, p_{13}\rangle, \langle p_4, p_{14}\rangle, \langle p_6, p_{10}\rangle, \langle p_7, p_{11}\rangle, \langle p_5, p_9\rangle\},$$

$$K_1' = \{\langle p, q\rangle : \langle q, p\rangle \in K_1\};$$

$$\mu_2 = \mu_1.$$

Hence we have $\mu_\infty = \mu_1$. The set P/μ_∞ has the following elements:

$$\{p_0\}, \quad \{p_1\}, \quad \{p_2, p_{12}\}, \quad \{p_3, p_{13}\}, \quad \{p_4, p_{14}\}, \quad \{p_5, p_9\}, \quad \{p_6, p_{10}\}, \quad \{p_7, p_{11}\}, \quad \{p_8\}.$$

The F-program and its partition $Q_0 = \{\hat{p}_0, \ldots, \hat{p}_8\}$ constructed on the basis of the last step of our procedure is given in Fig. 3. 1 is the address of this F-program.

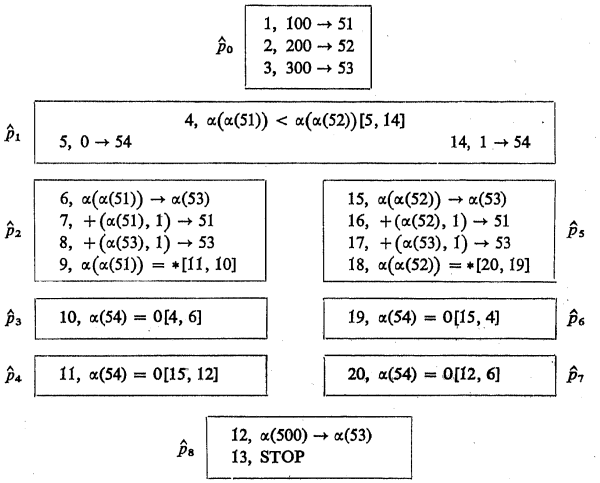


Fig. 3

5. Decomposibility of programs

In this section programs are treated as elements of partial algebras. The composition operation of programs is introduced. Composition determines in a natural way the notion of an indecomposable element. Necessary and sufficient conditions for the indecomposibility of a program and also a unique decomposibility of a program on indecomposable ones are given.

Let $M = \langle C, R, \varrho, \varkappa, \lambda, l\rangle$ be any SPC and let $c \in C$ be a memory state of M.

If $\varphi^c = \{\langle a, \varkappa(c(a))\rangle : a \in D(c)\} \cap (A \times R)$ is a finite set, then it will be called the *program of the state c*. Thus the program of a state c is a maximal set of labelled instructions connected in a natural way with the state c. Of course the program of any state c is a program. On the other hand, for any program φ there exists a state c such that $\varphi = \varphi^c$.

We assume in this and in the next sections that SPC satisfies the following conditions:

(i) $C = B^{[A]}$, (ii) $(\forall c)_C(\varphi^c = \varphi^{\pi(c)})$.

The adjoining of these conditions to be satisfied by SPC will prove very convenient in Section 6 of this Chapter. Now, we shall define a binary relation on $A \times R$ which is a kind of relation of a potential successor. For any $u_1, u_2 \in A \times R$ we adopt

$$u_1 \mathfrak{N} u_2 \quad \text{iff} \quad (\exists c_1, c_2)_C (\forall i)[\langle c_i(l), r_{c_i} \rangle = u_i \,\&\, \pi(c_1) = c_2].$$

The relation $\mathfrak{N}$ is a generalization of the usual relation of a successor, generated by a computation. More intuitively, it is a generalization of a step of the action of a machine. An advantage of this relation is its independence of machine computations and the fact that it is defined on the whole set $A \times R$.

The transitive closure of $\mathfrak{N}$ in φ we denote by D_φ.

Thus, for any $u_1, u_2 \in A \times R$

$$u_1 D_\varphi u_2 \quad \text{iff} \quad (\exists \{u^i\}_{i=1}^n \subset \varphi)(\forall i)(1 \leqslant i \leqslant n-1)[u^i \mathfrak{N} u^{i+1} \,\&\, u_1 = u^1 \,\&\, u_2 = u^n].$$

Let φ be an arbitrary program. A set $\varphi' \subset \varphi$ will be said to be a *set of generators of* φ provided that:

$$(\forall u)_\varphi (\exists u')_{\varphi'}[u' D_\varphi u].$$

An arbitrary program $\{u_i\}_{i=1}^n$ is a *cycle* if there exists a permutation σ of the sequence $(1, 2, \ldots, n)$ such that

$$(\forall i)(1 \leqslant i \leqslant n-1)[u_{\sigma(l)} \mathfrak{N} u_{\sigma(l+1)} \,\&\, u_{\sigma(n)} \mathfrak{N} u_{\sigma(1)}].$$

A program with a minimal set of generators whose number is equal to p will be called a *program of type p*.

Now we shall introduce the notion of a composition of programs, to be denoted by "$\odot$".

For any programs φ_1, φ_2 we assume that $\varphi_1 \odot \varphi_2$ is determined iff

(i) $\varphi_1 \cap \varphi_2 = \emptyset$;

(ii) $\varphi_1 \cup \varphi_2$ is a program;

(iii) if $\varphi \subset \varphi_1 \cup \varphi_2$ and φ is a cycle, then $\varphi \subset \varphi_1$ or $\varphi \subset \varphi_2$;

(iv) there exists a minimal set φ_2^0 of generators of φ_2 such that $\varphi_2^0 \subset \{u: (\exists u_1)_{\varphi_1}[u_1 \mathfrak{N} u]\}$.

If $\varphi_1 \odot \varphi_2$ is determined, then

$$\varphi_1 \odot \varphi_2 \overset{\text{df}}{=} \varphi_1 \cup \varphi_2.$$

Let $\{\varphi_k\}_{k \in K}$ be the family of all programs and $\mathscr{A}_k = \{\varphi: \varphi \subset \varphi_k\}$, $k \in K$.

We observe that instead of condition (ii) of the last definition we can put the following condition:

(v) $(\exists k)_K(\varphi_1 \in \mathscr{A}_k \,\&\, \varphi_2 \in \mathscr{A}_k)$.

The sets of programs $\mathscr{A}_k$, $k \in K$, will play an important part in the composition of computations to be considered in Section II.6.

A program φ will be called *decomposable* if there exists programs φ_1, φ_2 such that $\varphi = \varphi_1 \odot \varphi_2$.

In the opposite case we call a program φ an *atom*.

THEOREM 5.1. *Let φ be an arbitrary program of type* 1. *Then the following conditions are equivalent*:

(i) *φ is an atom,*

(ii) *D_φ is an equivalence relation on φ or* $\operatorname{card}(\varphi) = 1$.

Proof. Assume that φ is not an atom and that D_φ is an equivalence relation on φ. Hence there exist φ_1, φ_2 such that $\varphi = \varphi_1 \odot \varphi_2$. By the definition of operation "$\odot$" we obtain the existence of $u \in \varphi_1$, $u' \in \varphi_2$ such that $u \mathfrak{N} u'$. Since D_φ is a symmetric relation, we have $u' D_\varphi u$. Assume that $\{u_i\}_{i=1}^n \subset \varphi$ is a set defined by the formula

$$(\forall i)(1 \leqslant i \leqslant n-1)[u_i \mathfrak{N} u_{i+1} \,\&\, u' = u_1 \,\&\, u = u_n].$$

Clearly $\varphi_0 = \{u_i\}_{i=1}^n$ is a cycle. But $\varphi_0 \cap \varphi_1 \neq \emptyset$ and $\varphi_0 \cap \varphi_2 \neq \emptyset$, which contradicts the equality $\varphi = \varphi_1 \odot \varphi_2$.

Assume now that $\operatorname{card}(\varphi) > 1$ and φ is an atom. Let $u \in \varphi$. Since φ is a program of type 1, there exists a $u' \in \varphi$ such that $u D_\varphi u'$ or $u' D_\varphi u$. Suppose that $u D_\varphi u'$ and $\operatorname{non}(u' D_\varphi u)$. Hence $\varphi = (\varphi - \{u'\}) \odot \{u'\}$, which contradicts the fact that φ is an atom. If $u' D_\varphi u$ and $\operatorname{non}(u D_\varphi u')$, then we draw a similar conclusion. Thus $u D_\varphi u$. Let u, u' be arbitrary elements of φ such that $u D_\varphi u'$ and $\operatorname{non}(u' D_\varphi u)$. In that case $\varphi = (\varphi - \{u'\}) \odot \{u'\}$, which contradicts the fact that φ is an atom. Hence D_φ is a symmetric relation. By the definition of D_φ we infer that D_φ is a transitive relation on φ. Thus D_φ is an equivalence relation on φ. ■

THEOREM 5.2. *If φ is a program, then φ can be decomposed into atoms.*

Proof. Let φ be an arbitrary program and φ^0 its minimal set of generators. Let $\varphi_1 = \{u \in \varphi: (\exists u' \in \varphi^0)(u' D_\varphi u \,\&\, u D_\varphi u')\} \cup \varphi^0$. Clearly, φ_1 is an atom. If $\varphi_1 \neq \varphi$ then $\varphi = \varphi_1 \odot (\varphi - \varphi_1)$. Let $u_2 \in \varphi - \varphi_1$ fulfil the following condition: $(\exists u)_{\varphi_1}[u \mathfrak{N} u_2]$. We assume that $\varphi_2 = \{u \in \varphi: u_2 D_\varphi u \,\&\, u D_\varphi u_2\} \cup \{u_2\}$. By Theorem 5.1, we infer that φ_2 is an atom. Clearly $\varphi = \big((\varphi_1 \odot \varphi_2) \odot (\varphi - (\varphi_1 \cup \varphi_2))\big)$. In a similar manner we construct a sequence $\varphi_3, \varphi_4, \ldots, \varphi_n$ such that

$$\varphi = \big(\ldots((\varphi_1 \odot \varphi_2) \odot \varphi_3) \odot \ldots \odot \varphi_n\big)$$

where φ_i are atoms. ■

THEOREM 5.3. *The pair $(\mathscr{A}_k, \odot)$, $k \in K$, is a partial algebra. The least set of its generators is equal to the set of all atoms of $\mathscr{A}_k$.*

Proof. Let $\varphi \in \mathscr{A}_k$. By Theorem 5.2, we can present the program φ as a composition of some atoms. Suppose now that there exists a set of generators $\mathscr{Z}$ of $(\mathscr{A}_k, \odot)$ which is a proper subset of a set of all atoms of $\mathscr{A}_k$. Hence, there exists an atom φ_0 which does not belong to $\mathscr{Z}$. But φ_0 is an indecomposable program. Hence φ_0 cannot be generated by $\mathscr{Z}$. ■

THEOREM 5.4. *Every* SPC *determines uniquely a class $\{(\mathscr{A}_k, \odot)\}_{k \in K}$ of partial algebras.*

The proof follows from the construction of $(\mathscr{A}_k, \odot)$, $k \in K$. ■

THEOREM 5.5. *Let φ be an arbitrary program. Then the following conditions are equivalent*:

(i) *the program φ can be uniquely decomposed into atoms of type* 1,

(ii) *D_φ is a connective relation on φ.*

Proof. Let us observe that the power of a minimal set of generators of the program φ is equal to 1. To prove this fact assume that $u_1, u_2 \in \varphi_0$, where φ_0 is a minimal set of generators of φ. Hence $\sim(u_1 D_\varphi u_2)$ and $\sim(u_2 D_\varphi u_1)$ which contradicts (ii). On the other hand, for any φ_1, φ_2 such that $\varphi = \varphi_1 \odot \varphi_2$, φ_1 is a program of type $\geqslant 2$, which contradicts (i).

Suppose now that D_φ is not connective on φ. Hence there exist $u_1, u_2 \in \varphi$, such that $\sim(u_1 D_\varphi u_2)$ and $\sim(u_2 D_\varphi u_1)$. Let $\varphi_1 = \{u \in \varphi: u_1 D_\varphi u\}$ and $\varphi_2 = \varphi - \varphi_1$. We shall show that $\varphi = \varphi_2 \odot \varphi_1$. Evidently $\varphi_1 \cap \varphi_2 = \emptyset$ and $\varphi_1 \cup \varphi_2$ is a program.

Let $\varphi_0 \subset \varphi_1 \cup \varphi_2$ be a cycle such that $\varphi_0 \cap \varphi_1 \neq \emptyset$ and $\varphi_0 \cap \varphi_2 \neq \emptyset$. Hence there exist $u_3 \in \varphi_1 \cap \varphi_0$, $u_4 \in \varphi_2 \cap \varphi_0$ such that $u_3 D_\varphi u_4$ and $u_4 D_\varphi' u_3$. By the definition of the program φ_1 we infer that $u_4 \in \varphi_1$, which contradicts the fact that $u_4 \in \varphi_2$. Thus, for any cycle $\varphi_0 \subset \varphi_1 \cup \varphi_2$, $\varphi_0 \subset \varphi_1$ or $\varphi_0 \subset \varphi_2$. Since φ is a program of type 1, we infer that condition (iv) of the definition of operation "$\odot$" is satisfied.

Suppose now that $\varphi_3 = \{u \in \varphi: u_2 D_\varphi' u\}$ and $\varphi_4 = \varphi - \varphi_3$. In a similar manner we can show that $\varphi = \varphi_4 \odot \varphi_3$. Thus the program φ cannot be uniquely decomposed, which contradicts the assumption.

Let φ be an arbitrary program and D_φ a connective relation on φ. Let $\{u_0\}$ be a minimal set of generators of φ. Let us assume that $\varphi_1 = \{u: u_0 D_\varphi u \;\&\; u D_\varphi u_0\} \cup \{u_0\}$. Obviously the program φ_1 is an atom and $\varphi = \varphi_1 \odot (\varphi - \varphi_1)$. Let $u_1 \in \varphi - \varphi_1$ be an element satisfying the formula $(\exists u)_{\varphi_1}(u \mathfrak{N} u_1)$. We assume that

$$\varphi_2 = \{u: u_1 D_\varphi u \;\&\; u D_\varphi u_1\} \cup \{u_1\}.$$

Obviously the program φ_2 is an atom and

$$\varphi = \big((\varphi_1 \odot \varphi_2) \odot (\varphi - (\varphi_1 \cup \varphi_2))\big).$$

In a similar manner we construct the sequence $\varphi_3, \varphi_4, \ldots, \varphi_n$ such that

$$\varphi = (\ldots(\varphi_1 \odot \varphi_2) \odot \ldots \odot \varphi_{n-1}) \odot \varphi_n,$$

where φ_i are atoms.

Evidently the decomposition mentioned above is unique. ■

6. Decomposibility of computations

In Section II.5 the operation of a composition of programs was introduced. We found that every program can be decomposed into atoms. Necessary and sufficient conditions for the decomposability of a program into atoms of the type 1 were

formulated. In a natural way the question arises whether it is possible to introduce in an analogous manner an operation of a composition of computations in SPC and whether it is possible to find a connection between a decomposition of a computation and a decomposition of a program assigned to that computation in a suitable way. In order to solve the question in what way to associate a program with a computation we return to the conditions adjoined to the class SPC in Section II.5. Consider an arbitrary computation of SPC and two arbitrary states c_1, c_2. The first condition ensures that $\varphi^{c_1} = \varphi^{c_2}$. Thus a program of an arbitrary state of a computation can be interpreted as a program of that computation.

Let $c_0 = (c_0, c_1, \ldots)$ be any computation of M and $l(c_0) = m$. We say that c_0 is a *computation over an algebra* $(\mathscr{A}_k, \odot)$ if:

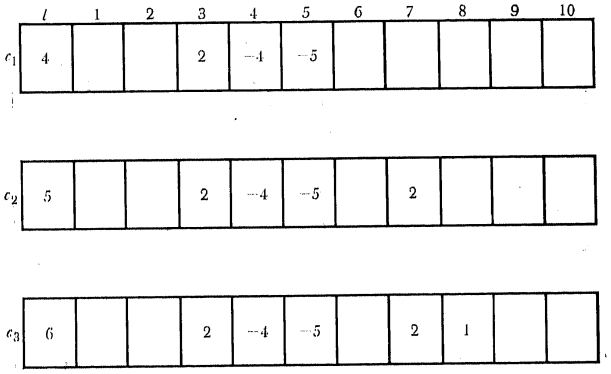
(i) $\varphi^{c_0} \in \mathscr{A}_k$,

(ii) $\bigcup_{i=0}^{m-1} \big(D(c_i) - D(\varphi^{c_0})\big) \cap D(\varphi_k) = \emptyset$.

EXAMPLE 6.1. Let M be an SPC such that A is a set of natural numbers, B is a set of integers and $\varkappa$: $B - A \to R$ is a function. We assume that

$$\varphi_k = \{\langle i, \varkappa(-i)\rangle: i \in \{4, 5, 6, 10\}\}.$$

Let us assume in the sequel that $\bar{c}_1 = (c_1, c_2, c_3)$ is a computation defined as follows:



We observe that the computation $\bar{c}_1$ is a computation over $(\mathscr{A}_k, \odot)$.

Now we are going to introduce the notion of an s-composition of any two computations.

Let us take two arbitrary computations

$$\bar{c}_1^1 = (c_1^1, c_2^1, \ldots, c_n^1), \quad \bar{c}_1^2 = (c_1^2, c_2^2, \ldots), \quad l(\bar{c}_1^2) = m$$

over an algebra $(\mathscr{A}_k, \odot)$. We assume that $c_1^1 \cdot \bar{c}_1^2$ is determined iff

(i) $\varphi^{c_1^1} \cap \varphi^{c_1^2} = \emptyset$;
(ii) $c_n^1 | \big(D(c_n^1) \cap D(c_1^2)\big) = c_1^2 | \big(D(c_n^1) \cap D(c_1^2)\big)$;
(iii) $\big(D(c_n^1) - D(c_1^2)\big) \cap \bigcup_{i=1}^{m} D(c_i^2) = \emptyset$.

If $\bar{c}_1^1 \cdot \bar{c}_1^2$ is determined then we put

$$\bar{c}_1^1 \cdot \bar{c}_1^2 = (c_1, c_2, \ldots)$$

where

(i) $c_i | D(c^l_{l-(j-1)(n-1)}) = \begin{cases} c_i^1 & \text{if} \quad j = 1 \,\&\, 1 \leqslant i \leqslant n, \\ c^2_{i-n+1} & \text{if} \quad j = 2 \,\&\, i \geqslant n+1; \end{cases}$
(ii) $c_i | \big(D(c_n^1) - D(c_1^2)\big) = c_n^1 | \big(D(c_n^1) - D(c_1^2)\big)$ for $i \geqslant n+1$;
(iii) $c_i | \big(D(c_1^2) - D(c_n^1)\big) = c_1^2 | \big(D(c_1^2) - D(c_n^1)\big)$ for $1 \leqslant i \leqslant n$;
(iv) $D(c_i) = \begin{cases} D(c_i^1) \cup \big(D(c_1^2) - D(c_n^1)\big) & \text{if} \quad i \leqslant n, \\ D(c^2_{i-n+1}) \cup \big(D(c_n^1) - D(c_1^2)\big) & \text{if} \quad i > n. \end{cases}$

EXAMPLE 6.2. Let M be an SPC and let $\bar{c}_1$ be a computation as considered in Example 6.1. We assume that $\bar{c}_4 = (c_4, c_5, c_6)$ is the following computation:

	l	1	2	3	4	5	6	7	8	9	10
c_4	6		4				−6			1	−10
c_5	10		2				−6			1	−10
c_6	4		2				−6			1	−10

Clearly $\bar{c}_4$ is a computation over an algebra $(\mathscr{A}_k, \odot)$ where $\varphi_k = \{\langle i, \varkappa(-i)\rangle : i \in \{4, 5, 6, 10\}\}$.

We observe that $\bar{c}_1 \cdot \bar{c}_4$ is determined. Therefore $\bar{c}_1 \cdot \bar{c}_4 = (c_7, c_8, c_9, c_{10}, c_{11})$ where

	l	1	2	3	4	5	6	7	8	9	10
c_7	4		4	2	−4	−5	−6			1	−10
c_8	5		4	2	−4	−5	−6	2		1	−10
c_9	6		4	2	−4	−5	−6	2	1	1	−10
c_{10}	10		2	2	−4	−5	−6	2	1	1	−10
c_{11}	4		2	2	−4	−5	−6	2	1	1	−10

This example shows that the s-composition of two computations is not always a computation. The second condition assumed additionally for the class SPC in Section II.5 ensures that the sequence of states obtained as a result of the s-composition of two computations is either a computation or the beginning of a computation.

Now we are going to introduce the notion of a composition of any two computations.

For any two computations $\bar{c}_1, \bar{c}_2$ we assume that:
$\bar{c}_1 \odot \bar{c}_2$ is determined iff $\bar{c}_1 \cdot \bar{c}_2$ is determined and $\bar{c}_1 \odot \bar{c}_2$ is a computation.
If $\bar{c}_1 \odot \bar{c}_2$ is determined then we put $\bar{c}_1 \odot \bar{c}_2 \stackrel{\text{df}}{=} \bar{c}_1 \cdot \bar{c}_2$.

THEOREM 6.1. *For any computations $\bar{c}_1^1, \bar{c}_2^1, \bar{c}_3^1$ over an algebra $(\mathscr{A}_k, \odot)$ the following two properties hold:*

(i) $\bar{c}_1^1 \odot (\bar{c}_2^1 \odot \bar{c}_3^1)$ *is determined if and only if* $(\bar{c}_1^1 \odot \bar{c}_2^1) \odot \bar{c}_3^1$ *is determined,*
(ii) *if* $\bar{c}_1^1 \odot (\bar{c}_2^1 \odot \bar{c}_3^1)$ *is determined then* $\bar{c}_1^1 \odot (\bar{c}_2^1 \odot \bar{c}_3^1) = (\bar{c}_1^1 \odot \bar{c}_2^1) \odot \bar{c}_3^1$.

Proof. Let $\bar{c}_1^1 = (c_1^1, c_2^1, \ldots, c_1^n)$, $\bar{c}_2^1 = (c_2^1, c_2^2, \ldots, c_2^m)$, $\bar{c}_3^1 = (c_3^1, c_3^2, \ldots)$ where $l(\bar{c}_3^1) = r$.

Let us assume that $(\bar{c}_1^1 \odot \bar{c}_2^1) \odot \bar{c}_3^1$ is determined where $(\bar{c}_1^1 \odot \bar{c}_2^1) \odot \bar{c}_3^1 = (c_4^1, c_4^2, \ldots, c_4^n, \ldots, c_4^{n+m-1}, \ldots, c_4^{n+m+s}, \ldots)$.

By the definition of a composition of any two computations over an algebra $(\mathscr{A}_k, \odot)$ we obtain

(i) $\big(D(c_1^n) - D(c_2^1)\big) \cap \bigcup_{i=1}^{m} D(c_2^i) = \emptyset$,

(ii) $\Big(\big(D(c_2^m) \cup \big(D(c_1^n) - D(c_2^1)\big)\big) - D(c_3^1)\Big) \cap \bigcup_{i=1}^{r} D(c_3^i) = \emptyset$,

(iii) $\big(D(c_2^1) - D(c_1^n)\big) \cap \bigcup_{i=1}^{n} D(c_1^i) = \emptyset$,

(iv) $\Big(D(c_3^1) - \big(D(c_2^n) \cup \big(D(c_1^n) - D(c_2^1)\big)\big)\Big) \cap \big(\bigcup_{i=1}^{m} D(c_2^i) \cup \bigcup_{i=1}^{n} D(c_1^i)\big) = \emptyset$.

From condition (ii) we obtain

$$\big(D(c_2^m) - D(c_3^1)\big) \cap \bigcup_{i=1}^{r} D(c_3^i) = \emptyset.$$

From condition (iv) we obtain

$$\big(D(c_3^1) - D(c_2^m)\big) \cap \bigcup_{i=1}^{m} D(c_2^i) = \emptyset.$$

From conditions (iii), (iv) we obtain

$$\Big(\big(D(c_2^1) \cup \big(D(c_3^1) - D(c_2^m)\big)\big) - D(c_1^n)\Big) \cap \bigcup_{i=1}^{n} D(c_1^i) = \emptyset.$$

From conditions (i), (ii) we obtain

$$\Big(D(c_1^n) - \big(D(c_2^1) \cup \big(D(c_3^1) - D(c_2^m)\big)\big)\Big) \cap \big(\bigcup_{i=1}^{m} D(c_2^i) \cup \bigcup_{i=1}^{r} D(c_3^i)\big) = \emptyset.$$

Hence and from the definition of a composition of any two computations over an algebra $(\mathscr{A}_k, \odot)$ we obtain

$$c_4^n|\big(D(c_1^n) - D(c_2^1)\big) = c_4^{n+m-1}|\big(D(c_1^n) - D(c_2^1)\big).$$

Thus the expression $\bar{c}_1^1 \odot (\bar{c}_2^1 \odot \bar{c}_3^1)$ is determined. Clearly,

$$(\bar{c}_1^1 \odot \bar{c}_2^1) \odot \bar{c}_3^1 = \bar{c}_1^1 \odot (\bar{c}_2^1 \odot \bar{c}_3^1). \blacksquare$$

A computation $\bar{c}$ is said to be *decomposable* if there exist computations $\bar{c}_1, \bar{c}_2$ such that $\bar{c} = \bar{c}_1 \odot \bar{c}_2$.

In the opposite case we say that $\bar{c}$ is an *atom*.

THEOREM 6.2. *An arbitrary computation can be decomposed into atoms.*

The proof follows from the fact that programs are finite sets. ■

Let $\bar{c}$ be any computation of an SPC. Observe that the statements:

(i) the decomposability of the computation $\bar{c}$ implies the decomposability of the program φ^c,

(ii) the decomposability of the program φ^c implies the decomposability of the computation $\bar{c}$,

are not always true.

The following question arises: is it possible to indicate a class of computations of an SPC satisfying (i), (ii)? Before answering this question we shall introduce an auxiliary notion.

For any computation $\bar{c}_1 = (c_1, c_2, \ldots)$, $l(\bar{c}_1) = n$, let

$$\mathfrak{B}(\bar{c}_1) = \bigcup_{i=1}^{n} \big(D_{c_i}(r_{c_i}) \cup Re_{c_i}(r_{c_i}) \cup c_i(\iota)\big) \cup \{\iota\}.$$

The set $\mathfrak{B}(\bar{c}_1)$ is said to be the *active field of the computation* $\bar{c}_1$.

Let $\bar{c}_1 = (c_1, c_2, \ldots)$, $l(\bar{c}_1) = n$ be any computation of SPC. Assume that:

$$\bar{c}_1 \in C_M^0 \quad \text{iff} \quad (\forall i \leqslant n)\big(c_i = c_i|\mathfrak{B}(\bar{c}_1)\big).$$

THEOREM 6.3. *The pair* $(C_M, \odot)$ *is a partial semigroup.*

The proof follows from Theorem 6.1 and the definition of the operation of composition. ■

THEOREM 6.4. *The pair* $(C_M^0, \odot)$ *is a partial subsemigroup of* $(C_M, \odot)$. *The least set of its generators is equal to the set of all atoms of* C_M^0.

Proof. If $\bar{c}_1, \bar{c}_2 \in C_M^0$ and $\bar{c}_1 \odot \bar{c}_2$ is determined, then $\bar{c}_1 \odot \bar{c}_2 \in C_M^0$. Hence and from Theorem 6.1 we obtain the proof of the first part of this theorem. The second part follows from Theorem 6.2. ■

THEOREM 6.5. *Let* $\bar{c}_1 = (c_1, c_2, \ldots)$, $l(\bar{c}_1) = n$ *be any computation of the set* C_M^0 *satisfying for every* $i \leqslant n$ *the condition*:

(i) $D(\varphi^{c_i}) \cap D_{c_i}(r_{c_i}) \subset \{c_i(l)\}$.

If $\varphi^{c_1} = \varphi_1 \odot \varphi_2$ *then there exist states* $c^1, c^2 \in C$ *such that* $\varphi_1 = \varphi^{c^1}$, $\varphi_2 = \varphi^{c^2}$ *and* $\bar{c}_1 = \bar{c}^1 \odot \bar{c}^2$.

Proof. Let c_i be a state of the computation $\bar{c}_1$ such that $c_i(l) \in D(\varphi_1)$ and $c_{i+1}(l) \notin D(\varphi_1)$. Such a state exists because $\bar{c}_1 \in C_M^0$. Let $c_1' = c_1|\big(D(c_1) - D(\varphi_2)\big)$ and $c_{i+1}' = c_{i+1}|\big(D(c_{i+1}) - D(\varphi_1)\big)$. We assume that $c^1 = c_1'|\mathfrak{B}(\bar{c}_1')$ and $c^2 = c_{i+1}'|\mathfrak{B}(\bar{c}_{i+1}')$.

We shall show that $c_{i+j}(l) \notin D(\varphi_1)$ for $j \geqslant 1$.

Let us assume that, for some $k \geqslant 1$, $c_{i+k}(l) \notin D(\varphi_1)$ and $c_{i+k+1}(l) \in D(\varphi_1)$. Since $c_{i+k}(l) \notin D(\varphi_1)$, we have $c_{i+k}(l) \in D(\varphi_2)$. Thus

$$\langle c_{i+k}(l), r_{c_{i+k}} \rangle \mathfrak{N} \langle c_{i+k+1}(l), r_{c_{i+k+1}} \rangle,$$

which contradicts the fact that $\varphi_1 \odot \varphi_2$ is determined. Hence and by (i) we obtain $\varphi_1 = \varphi^{c^1}$ and $\varphi_2 = \varphi^{c^2}$.

The equality $\bar{c}_1 = \bar{c}^1 \odot \bar{c}^2$ can be shown in an easy way. ■

Thus we have indicated a subclass of a class of computations of an SPC whose elements satisfy (ii). In order to solve the problem connected with (i) with every computation $\bar{c}$ of the stored program computer M we associate a relation $\mathfrak{N}_c \subset \varphi^c \times \varphi^c$.

Let $\bar{c}_1 = (c_1, c_2, \ldots)$, $l(\bar{c}_1) = n$ and $u_1 = \langle a_1, r_1\rangle$, $u_2 = \langle a_2, r_2\rangle \in \varphi^{c_1}$. We assume that

$$u_1 \mathfrak{N}_c u_2 \quad \text{iff} \quad (\exists i \leqslant n)[c_{i-1}(l) = a_1 \ \& \ c_i(l) = a_2].$$

THEOREM 6.6. *Let* $\bar{c}_1 = \bar{c}^1 \odot \bar{c}^2 \in C_M^0$ *and* $\mathfrak{N}_{c_1}|\varphi^{c_1} = \mathfrak{N}|\varphi^{c_1}$. *Then* $\varphi^{c_1} = \varphi^{c^1} \odot \varphi^{c^2}$.

Proof. The proof follows from the assumption $\varphi^{c^1} \cap \varphi^{c^2} = \emptyset$. Let $\varphi \subset \varphi^{c_1}$ be a cycle. Let us assume that there exist $\langle a_1, r_1\rangle \in \varphi^{c^1} \cap \varphi$, $\langle a_2, r_2\rangle \in \varphi^{c^2} \cap \varphi$ such that $\langle a_2, r_2\rangle \mathfrak{N} \langle a_1, r_1\rangle$. Hence $\langle a_2, r_2\rangle \mathfrak{N}_{c_1} \langle a_1, r_1\rangle$, which contradicts the fact that $\bar{c}^1 \odot \bar{c}^2$ is determined. ■

THEOREM 6.7. *Every computation in* C_M^0 *can be uniquely presented as a composition of atoms in* C_M^0.

Given an arbitrary computation $\bar{c}$, it can be decomposable into indecomposable parts. Every indecomposable part will be restricted to its active field. Theorem 6.7 shows a way of dividing the time of performing a computation into, in a sense, the least time sections. It shows also which part of the machine memory in these time sections is active.

7. Classification of programs

In this section a classification of computations of a stored program computer M and a classification of programs of M is proposed. In order to compare two arbitrary programs we should take under consideration their structures, the memory occupied by the computations of the programs and the lengths of those computations. Thus the comparison of programs should depend upon three parameters. On the other hand, taking under consideration three parameters, we should establish their importance. This may be done by introducing a weight function. It is clear that the choice of a method to compare programs using the assumptions mentioned above would involve us in rather long and arduous discussions. Thus we have chosen another way. Given a stored program computer M, with every program the set of its computations is associated. The comparison of programs is reduced to a comparison of computations.

We observe now that some computations compute the same things. So it is quite natural to introduce a strong congruence relation (see [10]) in $(C_M^0, \odot)$.

A congruence relation $\delta \subset C_M^0 \times C_M^0$ in $(C_M^0, \odot)$ is called *strong* if whenever $\bar{c}_1 \delta \bar{c}_2$, $\bar{c}_3 \delta \bar{c}_4$ and $\bar{c}_1 \odot \bar{c}_3$ is determined, then $\bar{c}_2 \odot \bar{c}_4$ is determined.

Let us assume that δ is a strong congruence relation in $(C_M^0, \odot)$.

We shall define a set of expressions C_M^δ and the partial operation $\odot_\delta$ on C_M^δ.

Let $\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n$ be arbitrary computations in C_M^0. We assume that:

$$\bar{c}_1 \odot_\delta \bar{c}_2 \odot_\delta \ldots \odot_\delta \bar{c}_n \in C_M^\delta \quad \text{iff} \quad (\forall i \leqslant n)(\exists \bar{c}_i' \in C_M^0)$$
$$[\bar{c}_1' \odot \bar{c}_2' \odot \bar{c}_3' \odot \ldots \odot \bar{c}_n' \in C_M^0 \ \& \ \bar{c}_i \delta \bar{c}_i'].$$

Let us observe that the symbol "$\odot_\delta$" can be interpreted as a partial operation on C_M^δ.

If $e_1, e_2 \in C_M^\delta$ where $e_1 = \bar{c}_1 \odot_\delta \bar{c}_2 \odot_\delta \ldots \odot_\delta \bar{c}_n$, $e_2 = \bar{c}_1' \odot_\delta \bar{c}_2' \odot_\delta \ldots \odot_\delta \bar{c}_m'$ then:

$e_1 \odot_\delta e_2$ is determined iff $\bar{c}_n \odot_\delta \bar{c}_1' \in C_M^\delta$.

If $e_1 \odot_\delta e_2$ is determined then we put

$$e_1 \odot_\delta e_2 = \bar{c}_1 \odot_\delta \bar{c}_2 \odot_\delta \ldots \odot_\delta \bar{c}_n \odot_\delta \bar{c}_1' \odot_\delta \bar{c}_2' \odot_\delta \ldots \odot_\delta \bar{c}_m'.$$

Thus we have obtained the partial algebra $(C_M^\delta, \odot_\delta)$. It is easy to verify that the least set of its generators is equal to C_M^0.

The question arises how to compare the elements of the algebra $(C_M^\delta, \odot_\delta)$. Of course we should define some 2-argument relation in C_M^δ. This relation ought to take into consideration the length of the computations and the structure of programs which are associated with computations. However, it does not need to take into consideration the memory occupied by computations, because C_M^0 is the set of generators of the partial algebra $(C_M^\delta, \odot_\delta)$. The definition of such a relation is not easy and would involve us in rather long discussions. We have chosen another way.

Let $e, e' \in C_M^\delta$. We assume that:

$$e \underset{\delta}{\subset} e' \text{ iff } e = \bar{c}_1 \odot_\delta \bar{c}_2 \odot_\delta \ldots \odot_\delta \bar{c}_n \ \& \ (\forall i \leqslant n)$$
$$[\bar{c}_i\text{—indecomposable} \ \& \ \bar{c}_i \delta \bar{c}_i'] \ \& \ e' = \bar{c}_1' \odot_\delta \bar{c}_2' \odot_\delta \ldots \odot_\delta \bar{c}_n'.$$

Let us note that the relation "$\underset{\delta}{\subset}$" is reflexive and transitive on C_M^δ. Thus $(C_M^\delta, \underset{\delta}{\subset})$ is a quasi-ordered set. We observe that this relation takes into consideration only the length of the computations.

Now we shall introduce the notion of a characteristic of a computation in C_M^0. At the very beginning we shall give the definition of a $\mathfrak{N}_c$-cycle.

An arbitrary program $\varphi = \{u_i\}_{i=1}^n$, where $\varphi \subset \varphi^c$, is said to be an $\mathfrak{N}_c$-*cycle* if there exists a permutation σ of the sequence $(1, 2, \ldots, n)$ such that

$$(\forall i)(1 \leqslant i \leqslant n-1)[u_{\sigma(i)} \mathfrak{N}_c u_{\sigma(i+1)} \ \& \ u_{\sigma(n)} \overline{\mathfrak{N}}_c u_{\sigma(1)}].$$

Let $\bar{c} \in C_M^0$. From Theorem 6.7 it follows that $\bar{c} = \bar{c}_1 \odot \bar{c}_2 \odot \ldots \odot \bar{c}_k$ where $\bar{c}_i \in C_M^0$ and $\bar{c}_i$ is an atom for each $i \in \{1, 2, \ldots, k\}$.

By n_i we denote the number of $\mathfrak{N}_{c_i}$-cycles in φ^{c_i}.

Consider a sequence $(h_1, h_2, \ldots, h_{k+1})$ such that:

(i) $h_1 = \operatorname{card}(\bigcup_{\substack{i=1 \\ n_i \neq 0}}^{k} \varphi^{c_i})$,

(ii) $(\forall i)(2 \leqslant i \leqslant k)[h_i \geqslant h_{i+1}]$,

(iii) $(h_2, h_3, \ldots, h_{k+1})$ is a permutation of the sequence $(n_1, n_2, \ldots, n_k)$.

The sequence $(h_1^*, h_2^*, \ldots, h_s^*)$ such that:

(i) $(\forall i \leqslant s)(h_i^* \neq 0 \ \& \ h_i = h_i^*)$,

(ii) $h_{s+1} = 0$

is said to be a *characteristic of the computation* $\bar{c}$.

THEOREM 7.1. *Every computation* $\bar{c} \in C_M^0$ *has exactly one characteristic.*

The proof follows from the definition of a characteristic. ■

Now, let H_M denote the set of characteristics of all computations in C_M^0. Clearly, for every stored program computer M, the set H_M is uniquely determined. Let us assume that H is the set defined as follows:

$h \in H$ iff there exist a stored program computer M and $\bar{c} \in C_M^0$ such that h is the characteristic of $\bar{c}$.

By "$\leqslant$" we denote the ordering relation on H which is defined as follows: for any $h_1 = (h_1^1, h_1^2, \ldots, h_1^n)$, $h_2 = (h_2^1, h_2^2, \ldots, h_2^m) \in H$, $h_1 \leqslant h_2$ iff one of the following conditions is satisfied:

(i) $(\exists s)(2 \leqslant s \leqslant \min\{m, n\})(\forall i)(2 \leqslant i < s)[(h_1^i = h_2^i) \,\&\, (h_1^s < h_2^s)]$,

(ii) $(\forall i)(2 \leqslant i \leqslant n)(h_1^i = h_2^i) \,\&\, [(n < m) \vee (n = m \,\&\, h_1^1 \leqslant h_2^1)]$.

THEOREM 7.2. *The pair $(H, \leqslant)$ is a well-ordered set.*

Proof. The empty sequence is the least element in $(H, \leqslant)$. From the definition of the relation "$\leqslant$" we infer that each two elements in H can be compared. Hence "$\leqslant$" is a connective relation on H. On the other hand, each element in H is isolated. Thus $(H, \leqslant)$ is a well-ordered set. ■

THEOREM 7.3. *Let M be any stored program computer. Then $(H_M, \leqslant)$ is a well-ordered set.*

The proof follows from Theorem 7.2.

According to the notation adopted in [12] the type of a linearly ordered set $(H, \leqslant)$ will be denoted by $\overline{H}$.

Every set $O_{\leqslant}(h) = \{h' \in H: h' \leqslant h \,\&\, h' \neq h\}$ will be called a *segment* of $(H, \leqslant)$.

Now we are going to introduce a two-argument operation on elements of the set H.

For any $h_1 = (h_1^1, h_1^2, \ldots, h_1^n)$, $h_2 = (h_2^1, h_2^2, \ldots, h_2^m)$, $h_3 = (h_3^1, h_3^2, \ldots, h_3^s) \in H$ we assume that $h_3 = h_1 \odot h_2$ iff the following conditions are satisfied:

(i) $h_3^1 = h_1^1 + h_2^1$;

(ii) the sequence $(h_3^2, h_3^3, \ldots, h_3^s)$ is a permutation of $(h_1^2, h_1^3, \ldots, h_1^n, h_2^2, h_2^3, \ldots, h_2^m)$.

THEOREM 7.4. *The pair $(H, \odot)$ is a commutative semigroup. The empty sequence is its zero element.*

The proof follows from the definition of the operation "$\odot$" on elements of the set H. ■

Let $h = (h_0, h_1, \ldots, h_k)$ be any element in H. Let s_i denote the least natural number which satisfies the inequality $h_i \leqslant 2^{s_i}$ where $i \in \{1, 2, \ldots, k\}$. Adopt the following notation $n_h \overset{\text{df}}{=} \sum_{i=1}^{k} s_i$.

The empty sequence will be denoted by e.

THEOREM 7.5. *Let M be an* SPC *and $h = (h_0, h_1, \ldots, h_k) \in H$. Then $h_0 \geqslant n_h$.*

Proof. Let $h \in H$. Then there exists a stored program computer M such that $h \in H_M$. Obviously, there exists a computation $\bar{c} \in C_M^0$ such that h is its characteristic. We can assume that $c = \bar{c}_1 \odot \bar{c}_2 \odot \ldots \odot \bar{c}_n$, where $\bar{c}_i \in C_M^0$ and $\bar{c}_i$ is an atom for every $i \in \{1, 2, \ldots, n\}$, $n \geqslant k$.

From the definition of a characteristic of any computation, we obtain $h_0 = \text{card}\,(\bigcup_{\substack{i=1 \\ n_i \neq 0}}^{n} \varphi^{c_i})$ where n_i denotes the number of $\mathfrak{N}_{c_i}$-cycles in φ^{c_i}.

Since the family $\{\varphi^{c_i}\}_{i \leqslant n}$ constitutes a partition of the program φ^c, we have $\text{card}\,(\bigcup_{\substack{i=1 \\ n_i \neq 0}}^{n} \varphi^{c_i}) = \sum_{\substack{i=1 \\ n_i \neq 0}}^{n} \text{card}\,(\varphi^{c_i})$.

The proof follows now from the fact that $n_i \leqslant 2^{s_i} - 1$, where $s_i = \text{card}\,(\varphi^{c_i})$ for $i \in \{1, 2, \ldots, n\}$, $n_i \neq 0$. ■

THEOREM 7.6. *Let $h = (h_0, h_1, \ldots, h_k) \in H$ where $h \neq e$. Then*

$$\overline{O_{\leqslant}(h)} = \omega^{h_1} + \omega^{h_2} + \ldots + \omega^{h_k} + (h_0 - n_h + 1).$$

The proof is obtain in an easy way.

COROLLARY 7.1. $\overline{H} = \omega^{\omega}$. ■

COROLLARY 7.2. *For any stored program computer M, $\overline{H}_M \leqslant \omega^{\omega}$.* ■

THEOREM 7.7. *Let h, h_1, h_2 be characteristics of computations $\bar{c}$, $\bar{c}_1$, $\bar{c}_2$, respectively. If $\bar{c} = \bar{c}_1 \odot \bar{c}_2$ then $h = h_1 \odot h_2$.*

Proof. Since $\bar{c} = \bar{c}_1 \odot \bar{c}_2$, we have $\varphi^{c_1} \cap \varphi^{c_2} = \emptyset$. Thus $h = h_1 \odot h_2$. ■

Let us observe that the relation $\underset{\delta}{\subset} \subset C_M^{\delta} \times C_M^{\delta}$ takes into consideration only the length of the computations. In order to consider the structure of programs which are connected with computations, we shall define a monotonic transfinite sequence $\{B_{M,\alpha}^0\}_{\alpha < \beta}$ of subsets of the set C_M^0, where $\beta \leqslant \omega^{\omega}$.

Let M be a stored program computer and let H_M be the set of characteristics of all computations in M. For every $\alpha < \overline{H}_M$ let us assume

$$\bar{c} \in B_{M,\alpha}^0 \quad \text{iff} \quad \overline{O_{\leqslant}(h)} \leqslant \alpha,$$

where h is the characteristic of $\bar{c}$.

It can be shown that

$$C_M^0 = \bigcup_{\alpha < \overline{H}_M} B_{M,\alpha}^0.$$

For every $\alpha < \overline{H}_M$ we define the set $B_{M,\alpha}^{\delta}$ on the basis of $B_{M,\alpha}^0$, just as the set C_M^{δ} was defined on the basis of C_M^0.

THEOREM 7.8. *For every $\alpha < \overline{H}_M$, $(B_{M,\alpha}^{\delta}, \odot_{\delta})$ is a subalgebra of $(C_M^{\delta}, \odot_{\delta})$.*

Proof. Let $e_1, e_2 \in B_{M,\alpha}^{\delta}$ and $e_1 \odot_{\delta} e_2$ is defined in $(C_M^{\delta}, \odot_{\delta})$. From the definition of $(B_{M,\alpha}^{\delta}, \odot_{\delta})$ we obtain $e_1 \odot_{\delta} e_2 \in B_{M,\alpha}^{\delta}$. ■

In order to consider the structure of programs which are connected with computations, a family $\{\underset{\delta,\alpha}{\subset}\}_{\alpha < \overline{H}_M}$ of binary relations defined on the set C_M^{δ} will be introduced.

For any $e_1, e_2 \in C_M^{\delta}$ we assume that $e_1 \underset{\delta,\alpha}{\subset} e_2$ iff the following two conditions are satisfied:

(i) $e_1 \underset{\delta}{\subset} e_2$,

(ii) α is the least ordinal number such that $e_1, e_2 \in B_{M,\alpha}^{\delta}$.

THEOREM 7.9. *The family* $\{\subset_{\delta,\alpha}\}_{\alpha<\bar{H}_M}$ *of relations constitutes a partition of the relation, i.e.,* $\subset_{\delta}$

(i) $\alpha \neq \beta \rightarrow \subset_{\delta,\alpha} \cap \subset_{\delta,\beta} = \emptyset$,

(ii) $\bigcup_{\alpha<\bar{H}_M} \subset_{\delta,\alpha} = \subset_{\delta}$.

The proof follows from the fact that for any α the relation "$\subset_{\delta,\alpha}$" is defined synonymously. ■

Thus, writing $e_1 \subset_{\delta} e_2$ for any elements $e_1, e_2 \in C^{\delta}_M$, we compare their lengths. It follows from Theorem 7.9 that there exists an ordinal number α such that $e_1 \subset_{\delta,\alpha} e_2$. This ordinal number determines the precision degree for the comparison of e_1, e_2 by the relation "$\subset_{\delta}$" with respect to the structure of programs connected with these elements. The smaller is the number α, the more precise is the comparison. The relation $\subset_{\delta,0}$ compares expressions in a precise way.

The question arises how to compare the programs of a stored program computer M.

Before answering this question we adopt the following notation.

Let $\bar{c} = \bar{c}_1 \odot \bar{c}_2 \odot \ldots \odot \bar{c}_n$ where $\bar{c}_i \in C^0_M$ are atoms. By $e(\bar{c})$ we shall denote the expression $\bar{c}_1 \odot_{\delta} \bar{c}_2 \odot_{\delta} \ldots \odot_{\delta} \bar{c}_n$.

Let φ be a program of a stored program computer M. Let $C_{M,\varphi}$ denote a subset of a set C^0_M which is defined as follows:

$$\bar{c} \in C_{M,\varphi} \quad \text{iff} \quad \varphi^c \subset \varphi \,\&\, \bar{c} \in C^0_M.$$

Now, for any programs φ_1, φ_2 of a stored program computer M we assume that $\varphi_1 <_{\delta,\alpha} \varphi_2$ iff the following three conditions are satisfied:

(i) $(\forall \bar{c}_1 \in C_{M,\varphi_1})(\exists \bar{c}_2 \in C_{M,\varphi_2})[\bar{c}_1 \,\delta\, \bar{c}_2]$,

(ii) $(\forall \bar{c}_2 \in C_{M,\varphi_2})(\exists \bar{c}_1 \in C_{M,\varphi_1})[\bar{c}_2 \,\delta\, \bar{c}_1]$,

(iii) $(\forall \bar{c}_1 \in C_{M,\varphi_1})(\forall \bar{c}_2 \in C_{M,\varphi_2})[(\bar{c}_1 \,\delta\, \bar{c}_2) \rightarrow (\exists \beta \leqslant \alpha)(\bar{c}_1 \subset_{\delta,\beta} \bar{c}_2)]$.

Thus with every program φ the set of computations $C_{M,\varphi}$ is associated. The comparison of programs is reduced to a comparison of computations.

References

[1] Amoroso, S., S. Bloom, *A model of the digital computer*, Symposium on Computers and Automata, Brooklyn 1971.

[2] Barański, H., *O obliczeniu w maszynie programowanej*, PWN, Warszawa 1972.

[3] Bartol, W., *On the existence of machine homomorphisms*, Bull. Acad. Polon. Sci., Sér. Sci. Math. Astr. Phys. 19 (1971), pp. 865–869.

[4] —, *On the existence of machine homomorphisms(II)*, ibid. 20 (1972), pp. 773–777.

[5] —, *Programy dynamiczne obliczeń*, PWN, Warszawa 1974.

[6] —, *Algebraic complexity of machines*, Bull. Acad. Polon. Sci., Sér. Sci. Math. Astr. Phys. 22 (1974), pp. 851–856.

[7] Cohn, P. M., *Universal algebra*, Harper and Row Publ., New York, Evanston and London 1965.

[8] Čulik, K., M.A. Arbib, *Sequential and jumping machines and their relation to computers*, Acta Inform. 2 (1973), pp. 162–171.

[9] Elgot, C.C., A. Robinson, *Random-access stored-program machines, an approach to programming languages*, Journ. ACM 11 (1964), pp. 365–399.

[10] Grätzer, G., *Universal algebra*, D. Van Nostrand Co. Inc. 1968.

[11] Kalmár, L., *Les calculatrices automatiques comme structures algébriques*, "Previsions, Calculs et Réalités", Paris 1965, pp. 9–22.

[12] Kuratowski, K., A. Mostowski, *Teoria mnogości*, PWN, Warszawa 1966.

[13] Kwasowiec, W., *Generable sets*, Inf. and Contr. 17 (1970), pp. 257–264.

[14] —, *Operations on relational machines*, Bull. Acad. Polon. Sci., Sér. Sci. Math. Astr. Phys. 18 (1970), pp. 765–768.

[15] Marica, J., S. Bryant, *Unary algebras*, Pacif. Journ. Math. 10 (1960), pp. 1347–1359.

[16] Mazurkiewicz, A. (Ed.), *Problemy przetwarzania informacji*, T. 2, Wyd. Nauk.-Techn., Warszawa 1974.

[17] Novotny, M., *O jednom problému z teorie zobrazení*, Spisy vyd. přírodoved. fakultou Masarykovy Univ. 344 (1953), pp. 53–64.

[18] —, *Über Abbildungen von Mengen*, Pacif. Journ. Math. 13 (1963), pp. 1359–1369.

[19] Pawlak, Z., *Maszyny programowane*, Algorytmy 10 (1969), pp. 7–22.

[20] —, *On the notion of a computer*, Logic, Meth. and Phil. of Science 3 (1968), pp. 255–267.

[21] —, *A mathematical model of digital computers*, Proc. Conf. Theory of Autom. and Formal Lang., Bonn 1973.

[22] Raś, Z., *On algebraic properties of computing machines*, Bull. Acad. Polon. Sci., Sér. Sci. Math. Astr. Phys. 18 (1970), pp. 613–618.

[23] —, *Congruence relations in computing machines*, ibid. 20 (1972), pp. 313–317.

[24] —, *On some properties of semiprograms in address machines*, ibid. 23 (1975), No. 2.

[25] —, *On some properties of computations in address machines*, ibid. 23 (1975), No. 2.

[26] —, *Classification of semiprograms of address machines*, ibid. to appear.

[27] Skowron, A., *On a certain class of stored program machines*, ibid. 21 (1973), pp. 859–865.

[28] —, *Algebraic properties of simulation*, CC PAS Reports, No. 183, 1975.

[29] Van Ba, N., *Semiprograms of address machines and their realizations*, Bull. Acad. Polon. Sci., Sér. Sci. Math. Astr. Phys. 19 (1971), pp. 1109–1115.

[30] —, *Subsemiprogram partitions of semiprograms of address machines*, ibid. 19 (1971), pp. 1117–1122.

[31] —, *ϱ-equivalent and ϱ-comorphic semiprograms of address machines*, ibid. 20 (1972).

[32] —, *The minimization problem for semiprograms of address machines*, ibid. 20 (1972).

[33] Wagner, E. G., *Bounded action machines: toward an abstract theory of computer structure*, Journ. of CSS 2 (1968), pp. 13–75.