DEMONSTRATIO MATHEMATICA
Vol. XXX No 4 1997

Waldemar Korczyński

ON A MODEL OF CONCURRENT SYSTEMS

Abstract. In some systems the possibility of concurrent execution of activities can be described by very irregular conditions which can not be modeled by Petri nets. In the paper a kind of models of such systems and a new definition of Petri nets are described. These models are graphs equipped with some families of sets representing actions of a system which can be executed independently. The presentation of graphs used in the paper allows for an unified treatment of various models of systems. There is also shown how some transformations e.g. refinement of concurrent systems can be modeled.

0. Introduction

0.1. Motivation. A concurrent system can be understood as a pair $S = (S, \rho)$ consisting of an object S and a set ρ of "rules". The object S, being a "static" part of such a system, describes *where* and the rules ρ *how* some processes may run. For example in Petri net theory of concurrence the object S is a Petri net, and the rules ρ describe which situations are permitted in S (e.g. capacity function for places) and how they can be changed (e.g. firing-rules). "Algebraising" such models of systems one defines them as some algebras. Now both, the object S (e.g. a Petri net) and the rules ρ (e.g. a specification of its behaviour) are described as an algebra $A=(A_1, \ldots, A_k, o_1, \ldots, o_m)$ satisfying some conditions (axioms). Such an algebraisation can consist e.g. of:

- a passing to quite different presentation (specification) with different signature. Typical examples are various passings from Petri nets to algebras (e.g. Montanari's & Mesequer's "monoids over graphs" [DDM91] or Winkowski's partial sequences [Win82]) .
- an enrichment of a relational system $S=(S, \rho)$ by some new operations. In this case the algebraisation consists of the introduction of some new operations and relations. These new operations (relations) describe some aspects of systems which can not be described by the "original" ones. Examples can

Key words and phrases: graph, concurrence, Petri net, refinement, system modeling.

be found e.g. in the theory of Mazurkiewicz's trace languages (the introduction of the congruence generated by the concurrence relation see e.g. [Maz77]).

• an axiomatization of a description of the system S=(S, ρ) as an algebra $(A_1, .., A_k, o_1, .., o_m)$. In this case systems are originally described by some algebras with an information about elements of their carrier sets (e.g. algebras of processes, algebras with carriers being multisets etc.). By an algebraisation one means here an axiomatic characterization of such class(es) of algebras. This is often used as the last part of algebraizations mentioned in the points 1 and 2 above.

The first approach leads usually to advanced category theory: in order to describe the relationship between the original and new presentation some complicated functors are used. In some cases the complication-degree of mathematical reasoning is higher than the complication-degree of considered systems. Another disadvantage of this way of "algebraisation" is the fact that often one has to compare (via corresponding functors) properties of quite different relational systems.

The second approach is in a sense more "classical". Introduction of new relations or operations into some structures is well known in mathematics. Typical examples are various ordered or topological structures (e.g. groups, rings, fields or vector spaces). It seems that it is easier to learn about some new properties of well known objects than about quite new objects and their relationship to the "old ones". In the paper a kind of such an approach is proposed. It is based on the fact that one of the best known formalisms used for the description of various dynamic systems is graph theory. Graphs are widely accepted as models of systems, not only in informatics, but in many other areas (e.g. economics or technology) as well. The idea of the proposed approach is to introduce into a graph a new structure describing a kind of synchronization and parallel composition of its elements. The mathematics used in this approach is very simple: it bases on some elementary notions of graph theory.

0.2. Models of concurrency. One of the most popular presentations of concurrent systems are graphs equipped with some additional structures (usually monoids) compatible with the graph-operations (e.g. graphs over monoids in [DDM91] or partially monoidal categories in [Kor80], [Win82]). In such an approach a system is seen as a pair S=(G, ρ) with G being a graph and ρ being the structure mentioned above. The carrier set of such a structure can be understood as a family of sets (usually called *steps*) of activities which can be executed independent each other. One of fundamental assumptions about such families is that they are downwards closed

under set inclusion i.e. if a set α (of activities) is allowed in a system, say S, then every of its subsets $\beta \subseteq \alpha$ is allowed in S as well. This is often a consequence of the fact that concurrence is understood as a *binary relation* in a set (e.g. of transitions of a Petri net). However there are systems in which this assumption is not satisfied. Examples can be found in practically every area (e.g. "minimal transactions", various "threshold phenomena" in physics etc.). In the paper we propose a simple formalism for describing such systems. The idea is very simple; some classes of algebras (e.g. semigroups) have not only HSP (i.e. they are closed wrt. the operation of forming homomorphic image, subalgebras and product), but are closed wrt. the operation of forming of a "power algebra" as well. One of such classes is the class of graphs. We show how a *family of independent sets* of activities can be defined as a subgraph of a "power graph", define a kind of morphisms of such systems and show some properties of corresponding categories. Considering a family of sets instead of a binary relation is well known in mathematics. Typical examples can be found in algebra where the principle of abstraction allows for a "translation" of some propositions formulated in the "language of binary relations" (equivalencies) into the "language of families of sets" (partitions). The idea to see concurrence as a family of sets is not new; it can be found in Petri's papers on net theory (see [Pe73]). The approach proposed in the paper differs from those related to "classical" Petri nets in two points:

- we consider a one sorted presentation of graphs and consequently sets consisting of elements of the same type i.e. in net terminology our "steps" consist of transitions *and* places
- the conditions postulated to be satisfied by the family of such "steps" are weaker than those postulated e.g. for the family of configurations in a Petri net.

0.3. Organization of the paper. The paper is organized as follows. In section 1 the notation and terminology used in the paper are described. In the paper some not very popular notions (e.g. a one-sorted presentation of graphs) are used. This section is devoted to a description of these notions. In section 2 we define the main kind of objects used in the paper; graphs with limitations. In section 3 a definition of morphisms of graphs with limitations is formulated and some of their properties (e.g. a construction of products and coproducts) are shown. Section 4, where some special morphisms of graph with limitations are considered, is in a sense a continuation of section 3. We describe here morphisms respecting processes in systems. A sketch of a definition of processes, which seems to be very natural in the proposed formalism is given in section 5 where we also show an example of a refinement

of a system by means of the introduced morphisms. In section 6, some general remarks about introduced notions are formulated.

1. Notation and terminology

In the paper the standard mathematical notation and terminology are used. Some not very often used notions are described below.

By a *graph* we mean any triple G=(X, dom, cod) such that X is a set and dom, cod: $X \to$X are unary operations satisfying the conditions:

$$\mathrm{dom}(\mathrm{dom}(x)) = \mathrm{cod}(\mathrm{dom}(x)) = \mathrm{dom}(x)$$
$$\mathrm{dom}(\mathrm{cod}(x)) = \mathrm{cod}(\mathrm{cod}(x)) = \mathrm{cod}(x)$$

for every $x \in X$. The passing from this definition to the two-sorted presentation of graphs as algebras of the form $G = (V, A, d_0, d_1)$ with V=Vertices(G) being the set of *vertices* of the graph G, A=Arrows(G) being the set of its *arrows* and functions d_0, d_1: A $\to$ V is via equations:

$$V = \mathrm{Vertices}(G) = \{x \in X : \mathrm{dom}(x) = \mathrm{cod}(x) = x\},$$
$$A = \mathrm{Arrows}(G) = X \setminus V, \quad d_0 = \mathrm{dom} \setminus \mathrm{id}_v, \quad d_1 = \mathrm{cod} \setminus \mathrm{id}_v .$$

We choose the one-sorted presentation of graphs because in this case, unlike e.g. programming, it is easier to work with one- than with two-sorted algebras. The class of morphisms of such graphs is richer than the class of "classical" homomorphisms of graphs. The "classical" homomorphisms of graphs as two sorted algebras map always vertices onto vertices and arrows (edges) onto arrows. This property makes impossible using them as models of any "aggregation" when an arrow should be mapped onto a vertex. This can be "improved" by considering another, not "typically algebraic", definition of graph morphisms. By such a morphism one understands e.g. any triple $f : G \to G'$ with G and G' being graphs and

$$f : \mathrm{Arrows}(G) \cup \mathrm{Vertices}(G) \to \mathrm{Arrows}(G') \cup \mathrm{Vertices}(G')$$

being a function satisfying the condition

$$\forall_{a \in \mathrm{Arrows}(G)}(f(d_0(a)) = d_0'(f(a)) \ \&\ f(d_1(a)) = d_1'(f(a))) \quad \text{or}$$
$$f(a) = f(d_0(a)) = f(d_1(a)).$$

An analogous approach can be found in some papers on Petri Nets (see e.g. definitions of net morphisms in [GLT]). This "improved" definition of graph morphism does not allow to use the Birkhoff's theorem. So the existence of products, homomorphic images and subalgebras has to be proved. If one wants to use graph homomorphisms as a model of a kind of "aggregation of

arrows into vertices" one has to choose between the simple two-sorted definition of graphs and complicated definition of their morphisms, or a "not typical" definition of graphs (in this case of the above one-sorted presentation) and typical algebraic definition of their morphisms. In the paper the second possibility has been chosen. For every graph $G = (X, \mathrm{dom}, \mathrm{cod})$ by a path we mean any (including empty) sequence $\mathbf{p} = a_1 a_2, \ldots, a_k$ of arrows of G with $\mathrm{dom}(a_i) = \mathrm{cod}(a_{i-1})$ for $1 \leq i < k$. For a path $\mathbf{p} = a_1 a_2, \ldots, a_k$ we define $\partial_0(p) = \mathrm{dom}(a_1)$ and $\partial_1(p) = \mathrm{cod}(a_k)$. The free category generated by G

$$\mathrm{Path}(G) = (V(G), \mathrm{Paths}(G), \partial_0, \partial_1, \bullet)$$

consists of:

- the set V(G) of all vertices of the graph G being the class of objects of the category Path(G),
- the set Paths(G) of all paths of the graph G being the class of morphisms of the category Path(G),
- ∂_0 and ∂_1 — the operations of the domain and codomain of the category Path(G) defined above and
- the operation "$\bullet$" of concatenation (of paths) restricted to the set

$$\mathrm{Dom}(^\circ) = \{(\mathbf{p}, \mathbf{q}) : \mathbf{p}, \mathbf{q} \in \mathrm{Paths}(G) \ \& \ \partial_1(\mathbf{p}) = \partial_0(\mathbf{q})\}$$

being the composition of morphisms of the category Path(G).

The insertion of generators is given by the embedding

$$\iota : \mathrm{Arrows}(G) \to \mathrm{Paths}(G) \quad \text{with} \quad \iota : \alpha \to \alpha.$$

One of very often used notions in various graph-theoretic models of systems and projects is the notion of consistence called in this paper independence (e.g. of fronts of activities). It can be defined in many ways; one of the most popular ones is the definition based on the notion of (algebraic) closure operation.

EXAMPLE 1.1. *Perhaps the best known example of independence is the linear independence of vectors. A vector v is independent on the set A iff it can not be written as a linear combination of vectors from the set A, i.e. if it can not be expressed by these vectors by means of the operations of addition of vectors and multiplication of vectors by scalars.*

The so understood independence can be generalized to a one-argument relation in the power set of a given set A as follows: a subset $A_0 \subseteq A$ of a set A is *independent* iff every element $x \in A_0$ is independent on the set $A_0 \backslash \{x\}$. (We assume the relation to be independent on a set which has been already defined). This is one of the most often used ways of defining the

linear independence of subsets in a linear space. The independence relation so obtained can be seen as a family $\mathrm{ind}(A) \subseteq \mathrm{Pow}(A)$ of subsets of a set A with the following property.

A set A_o *belongs to* $\mathrm{ind}(A)$ *iff every finite subset of* A_o *is independent.*

The above property is often used as a definition of independence. An independence *relation* (called independence in the sense of Marczewski see e.g. [Cohn68]) in a set A is any family $\mathrm{ind} \subseteq \mathrm{Pow}(A)$ such that for every $X \in \mathrm{Pow}(A)$

$$X \in \mathrm{ind} \Leftrightarrow \text{ for every finite subset } X_0 \subseteq X, X_0 \in \mathrm{ind}\,.$$

An *independence space* is any pair $\mathbf{I} = (A, \mathrm{ind})$ such that A is a set and *ind* an independence relation in A. Sets belonging to the class *ind* are called *independent* and all the other *dependent.*

Remark 1.1. In order to apply the above notion of independence in a concrete case one has to know when a finite subset $X_0 \subseteq A$ is independent. Such a notion of finite independent sets must be defined separately. In algebra one assumes that a finite set $X_0 = \{x_1, x_2, \ldots, x_k\}$ is dependent (in an algebra A) iff one of its elements can be expressed by means of operations of this algebra by other ones, i.e. X_0 is dependent iff there exists $j \leq k$ such that $x_j = \varphi(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_k)$ for an algebraic function (a polynomial) φ of the algebra A. Another possibility is to fix a family $\mathbf{D}$ of finite subsets of the set A and define a set to be dependent iff it contains an element of $\mathbf{D}$.

The so understood independence spaces are structures similar to topological spaces (are of "topological type" in the sense of Bourbaki). Their morphisms are defined similar to open (continuous) mappings of topological spaces. A triple $f : \mathbf{I} \to \mathbf{I}'$ with $\mathbf{I} = (A, \mathrm{ind})$, $\mathbf{I}' = (A', \mathrm{ind}')$ being independence spaces and $f : A \to A$ being a function is a (twisted) *morphism* from $\mathbf{I}$ to $\mathbf{I}'$ iff

$$f(\mathrm{ind}) \subseteq \mathrm{ind}'(f^{-1}(\mathrm{ind}') \subseteq \mathrm{ind} \text{ respectively}).$$

Example 1.2. *Typical examples of twisted morphisms of independence space are linear mappings; the counter image of a linear independent set is always linear independent.*

Remark 1.2. Such an approach to independence is not always so natural as in the above examples. Sometimes it is more convenient to see an independent set as a minimal set generating a subobject of an object. In such situations one can define independent sets in a set A as minimal elements of the quotient set consisting of equivalence classes of a congruence of the semilattice $(\mathrm{Pow}(A), \cup)$. In general, one needs an additional condition

guaranteeing that such minimal elements exist. Such a condition can be the finiteness of the set A (see e.g. [NP91] for more details). The differences in the both approaches can be illustrated on the well known example of the notion of a basis of a linear space. A basis can be defined as maximal independent set or as minimal generating set of a given linear space.

The reader is assumed to be familiar with the elementary notions of the theory of Petri nets.

2. Power graphs

It is well known that every function $f : X \to Y$ can be "extended" to the power function $\mathrm{pow}(f) : \mathrm{pow}(X) \to \mathrm{pow}(Y)$. This extension (being in fact a functor) is defined by the assignment

$$X \supseteq A \to \{y : \exists_{x \in A} y = f(x)\} = \mathrm{pow}(f)(A) \subseteq Y.$$

This functor assigns to every algebra A of a given signature σ a "power algebra" $\mathrm{pow}(A)$ of A with operations defined as the corresponding extensions of the operations of the original algebra. Unfortunately the assignment

$$A \to \mathrm{pow}(A)$$

usually loses many important properties of the original algebra and consequently the result algebra $\mathrm{pow}(A)$ has with the algebra A only the signature in common. Usually we are interested in some subalgebras of the algebra $\mathrm{pow}(A)$. Perhaps the most often case is that we are interested in subalgebras with carriers being partitions of corresponding carrier sets of original algebras. Such algebras are known as "quotient algebras". In the case of graphs the "power functor" leads to graphs, i.e. the power algebra of a graph is a graph as well. In this section we describe the corresponding construction and some of the properties of such algebras.

DEFINITION 2.1 (Power graph). *The power graph of a graph* $G = (X, \mathrm{dom}, \mathrm{cod})$ *is the algebra* $\mathrm{pow}(G) = (\mathrm{pow}(X), \mathrm{DOM}, \mathrm{COD})$ *where for every* $A \subseteq X$ *it holds*

$$\mathrm{DOM}(A) = \bigcup_{x \in A} \{\mathrm{dom}(x)\} \text{ and } \mathrm{COD}(A) = \bigcup_{x \in A} \{\mathrm{cod}(x)\}.$$

In the sequel we usually omit the brackets "{" and "}" in the expressions $\bigcup_{x \in A}\{\mathrm{dom}(x)\}$ and $\bigcup_{x \in A}\{\mathrm{cod}(x)\}$. It shouldn't cause mistakes because the meaning of these expressions will be always clear from the context.

PROPOSITION 2.1. *The power graph* $\mathrm{pow}(G)$ *of a graph* G *is a graph.*

In order to define a concurrent system as a pair of the form $S = (G, \sigma)$ we define in the carrier set X of the graph $(G = (X, \mathrm{dom}, \mathrm{cod})$ an independence

relation i.e. a family ind_G of subsets of X satisfying the conditions mentioned above. It can be described similar to the definition of the (in)dependence in logic. One has to define a family $\mathbf{D_0}$ of dependent sets and to put a set $A \subseteq \mathrm{Pow}(X)$ to be dependent iff it contains an element of the family $\mathbf{D_0}$. In order to obtain in this way a model of a concurrent system one also has to interpret the independent subsets of the carrier set X of a graph as some processes. Next one can define operations on them which correspond to some operations on processes in the system. The most natural way of defining such operations seems to be an extension of graph operations to the family of independent subsets of a graph. Now we can define a graph with (in)dependence as a triple of the form $G = (X, \mathrm{dom}, \mathrm{cod}, ind_G)$ such that $\mathrm{ind}_G \subseteq \mathrm{Pow}(X)$, and the pair (X, ind_G) is an independence system. If the family ind_G is closed wrt operations DOM and COD then the triple $(ind_G, \mathrm{DOM}, \mathrm{COD})$ is a graph. In this way we obtain a subgraph

$$\mathrm{ind}(G) = (\mathrm{ind}_G, \mathrm{DOM}, \mathrm{COD})$$

of the power graph $\mathrm{pow}(G)$ with elements being independent subsets of G. The above constructions can be easily generalized to the case when instead of independence relation ind_G one uses an arbitrary family of sets closed wrt the operations DOM and COD.

DEFINITION 2.2 (Limitation family). *A family $L \subseteq \mathrm{Pow}(X)$ is a limitation family of a graph $G = (X, \mathrm{dom}, \mathrm{cod})$ if the triple $(L, \mathrm{DOM}_{|L}, \mathrm{COD}_{|L})$ is a subgraph of the graph $\mathrm{pow}(G)$ i.e. iff $\mathrm{DOM}(X_o) \in L$ and $\mathrm{COD}(X_o) \in L$ for every $X_o \in L$. A limitation family L of a graph G is called safe iff for every $X_o \in X$*

$$(\mathrm{DOM}(X_o) \in L \text{ or } \mathrm{COD}(X_o)t \in L) \Rightarrow X_o \in L.$$

Few words of explanation can be helpful in understanding the definition. The elements of the limitation family L can be seen as sets of activities (represented by arrows of the corresponding graph) which can be executed in parallel or (if the corresponding set is minimal in the poset $(L, \subseteq)$) have to be executed synchronously. Such sets are often called *steps* or *fronts of activities*. In the "standard" approach to concurrence, steps in a system constitute an independence relation; a set (of some elements of the system e.g. transitions andór places in a Petri net) is a step iff each of its finite subsets is a step. Often even a "stronger" definition is used; a set, say A is a step iff any of its two-element subsets is a step (e.g. every two transitions belonging to A are concurrent). The main difference between our approach and the "standard" one is that limitation may be not an independence relation, even not downwards closed under set inclusion (down directed) i.e.

it is possible that it holds $A \subseteq B \;\&B \in L \;\&A \notin L$ for some subsets A and B of the carrier set of the corresponding graph. Such steps may represent nonsteps in a Petri-net. This property allows for a modeling of systems, which are more general then Petri nets. Safety of a concurrent system can be understood as the property saying that if a process within it starts from (ends in) a situation which is permitted in the system then this process is permitted in the system as well. This property is equivalent to the "classical", net-theoretic meaning of the safety; if some cuts of processes, say α and β can occur independent each other (are coexistent), then these processes can run parallel. If we want to see every process in a system as built up from some "steps" then it is enough to postulate such a property for these steps only. The condition

$$(\mathrm{DOM}_{|L}(X_0) \in L \text{ or } \mathrm{COD}_{|L}(X_0) \in L) \Rightarrow X_0 \in L$$

just means that if the start- or end-situation of a step X_0 is permitted in the system then the step X_0 is permitted in it as well.

PROPOSITION 2.2. *For every graph* $G = (X, \mathrm{dom}, \mathrm{cod})$ *the whole family* $\mathrm{Pow}(X)$ *and the empty family* $\emptyset$ *are safe limitations of* G.

DEFINITION 2.3 (Graph with limitation). *By a graph with limitation (L-graph for shortness) we mean any quadruple of the form* $\mathbf{GL} = (X, \mathrm{dom}, \mathrm{cod}, L)$ *such that the reduct* $\mathbf{graph}(\mathbf{GL}) = (X, \mathrm{dom}, \mathrm{cod})$ is a graph and L is a limitation family of it. A graph with limitation $\mathbf{GL} = (X, \mathrm{dom}, \mathrm{cod}, L)$ will be called complete iff $\bigcup L = X$.

From now on we will use complete graphs with limitations only.

An interpretation. An L-graph can be seen as a model of a system consisting of some atomic situations (represented by the vertices of the corresponding graph) and some atomic activities (represented by the arrows of this graph). These situations and activities can occur (be executed) only as sets of the family L.
The definition of limitation family is not a constructive one; it does not include any way of defining such families. The following properties of limitations allow to construct limitations by means of some operations on families of sets.

PROPOSITION 2.3. *The intersection of any family of limitations of a graph is a limitation of this graph. The intersection of a family of safe limitations is safe as well.*

COROLLARY 2.1. *The family of all limitations of a graph* G *constitutes a complete lattice with meet being the standard set theoretic intersection. The safe limitations constitute a sublattice of it.*

COROLLARY 2.2. *For every graph* $G = (X, \mathrm{dom}, \mathrm{cod})$ *and any family* $\mathbf{L} \subseteq \mathrm{Pow}(X)$ *there exists the last limitation* $L(\mathbf{L})$ *of the graph* G *containing the family* $\mathbf{L}$. *It will be called generated by* $\mathbf{L}$.

The above results allow to define limitations of graphs generated by some families of sets like topologies in a set. We use them to define limitations in some constructions on L-graphs (product and coproduct),

3. Some categories of graphs with limitations

3.1. Morphisms of graphs with limitation. One of the most significant steps in the development of the theory of concurrent systems was the "refinement" of steps into some parts. Steps have been treated as **sets** of some "more atomic" elements called e.g. **places** and **transitions** (**conditions** and **events**) in the case of Petri nets. Corresponding to this change of seeing systems the notion(s) of their allowed transformations have been changed. Such transformations, called *morphisms*, assign to systems some other, from a point of view "better" systems. They (i.e. transformations) are used as tools for designing or analyzing systems. In the case of sequential systems the situation is simple; systems are modeled by graphs, their transformations by graph-homomorphisms. There exists no problem with respecting or reflecting processes. A process in a sequential system is modeled by a path of the corresponding graph (the model of this system). Graph homomorphisms respect paths of graphs and consequently they respect the processes of a system as well. In the case of concurrent systems the problem of respecting processes has been in a sense (partially) pushed down onto the level of sets of arrows and vertices of a graph. Morphisms of corresponding systems can be defined as morphisms of power graphs respecting or reflecting their limitations. The appropriate definition could be e.g. of the following form.

DEFINITION 3.1.1 (Morphisms of graphs with limitation). *A triple* $f : \mathbf{GL} \to \mathbf{GL}'$ *with* $GL = (X, \mathrm{dom}, \mathrm{cod}, L)$, $GL = (X', \mathrm{dom}', \mathrm{cod}', L')$ *being L-graphs and* $f : L \in L'$ *being a function is called*:

(a) *a morphism of* **GL** *into* **GL'** (*L-morphism*) *iff*

$$f(\mathrm{DOM}(X_0)) \subseteq \mathrm{DOM}'(f(X_0)) \ \& \ f(\mathrm{COD}(X_0)) \subseteq \mathrm{COD}'(f(X_o))$$

for every $X_0 \in L$.

(b) *a twisted morphism of* **GL** *into* **GL'** (*TL-morphism*) *iff*

$$f^{-1}(\mathrm{DOM}'(X_o)) \subseteq \mathrm{DOM}(f^{-1}(X_0)) \ \& \ f^{-1}(\mathrm{COD}'(X_0)) \subseteq \mathrm{COD}(f^{-1}(X_0))$$

for every $X_0 \in L'$.

Remark 3.1.1. Morphisms of L-graphs in general **are not graph morphisms** satisfying some additional conditions (e.g. reflecting of limitation). They are defined as mappings of some families of sets. Considered as correspondences between graphs they are rather relations then functions.

Remark 3.1.2. Let us note; morphisms of L-graphs *respect* their steps and situations (i.e. a step in **GL** is transformed onto a step in **GL'**) and twisted morphisms *reflect* the steps (i.e. the counter-image $f^{-1}(A')$ of the step A' in **GL'** is always a step in **GL**).

The categories of L-graphs with simple and twisted morphisms will be denoted by the symbols GLGRAPH and TGLGRAPH respectively.

3.2. Products and coproducts. Both categories GLGRAPH and TGLGRAPH have products. The construction of product of L-graphs (G, L) and (G', L') in the category GLGRAPH is standard; the corresponding limitation family consists of sets of the form $A \times A'$ with $A \in L$ and $A' \in L'$.

PROPOSITION 3.2.1. *For every L-graphs* (G, L), (G', L') *and* (G'', L'') *and every morphisms* $f : G'', L'') \to (G, L)$ *and* $g : (G'', L'') \to (G', L')$ *of the category GLGRAPH*:

(a) *the family* $L \times L' = \{A \times A' : A \in L \ \& \ A' \in L'\}$ *is a limitation family of the product graph* $G \times G'$. *If both limitations are safe then so is the family* $L \times L'$.

(b) *the triple* $\langle f, g\rangle : (G'', L'') \to (G \times G', L \times L')$ *is a morphism of this category.*

(c) *the projections* $\mathrm{pr}_1 : G \times G', L \times L') \to (G, L)$ *and* $\mathrm{pr}_2 : G \times G', L \times L') \to G', L')$ *are morphisms of the category GLGRAPH.*

COROLLARY 3.2.1. *For every L-graphs* (G, L) *and* (G', L') *the pair* $(G \times G', L \times L')$ *is their product in the category GLGRAPH.*

PROPOSITION 3.2.2. *In the category GLGRAPH the coproduct of L-graphs* $\mathbf{GL} = (G, L)$ *and* $\mathbf{GL'} = (G', L')$ *is the pair* $\mathbf{GL} + \mathbf{GL'} = (G + G', L + L')$ *with* $L + L'$ *being the family*

$$L + L' = \{A + A' : A \in L \ \& \ A' \in L'\}.$$

The construction of products and coproducts in the category TGLGRAPH is a bit more complicated. The idea is similar to the construction of the product in the category of topological spaces i.e. one defines it as the product of corresponding graphs with limitation being the last family for which all projections reflect the limitations of the "factors".

PROPOSITION 3.2.3. *In the category TGLGRAPH the product of L-graphs* $\mathbf{GL} = (G, L)$ *and* $\mathbf{GL}' = (G', L')$ *is the pair* $\mathbf{GL} \times \mathbf{GL}' = (G \times G', \mathbf{L}(L, L'))$ *with* $\mathbf{L}(L, L')$ *being the last limitation for which both projections* $\mathrm{pr}_i : G \times G' \to G$ $(i = 1, 2)$ *are TGLGRAPH -morphisms.*

Proof. From Definition 3.1.1. we infer that

$$L \in \mathbf{L}(L, L')) \Leftrightarrow$$
$$\mathrm{DOM}\,\mathrm{pr}_1(A)) \subseteq \mathrm{pr}_1(\mathrm{DOM}(A)) \ \& \ \mathrm{DOM}(\mathrm{pr}_2(A)) \subseteq \mathrm{pr}_2(\mathrm{DOM}(A)).$$

The family $L_{G\times G'}$ of all limitation for which both projections are TGLGRAPH-morphisms is not empty because $\mathrm{Pow}(G \times G') \in \mathbf{L}(L, L')$ (for the simplicity we write here $\mathrm{Pow}(G \times G')$ instead of the more precise expression $\mathrm{Pow}(\mathit{carrier}(G) \times \mathit{carrier}(G'))$. The family $\mathbf{L}(L, L')$ is the intersection of the family $L_{G\times G'}$ and by Proposition 2.3. a limitation of the product-graph $G \times G'$.

PROPOSITION 3.2.4. *In the category TGLGRAPH the coproduct of L-graphs* $\mathbf{GL} = (G, L)$ *and* $\mathbf{GL}' = (G', L')$ *is the pair* $\mathbf{GL} + \mathbf{GL}' = (G + G', L \ddagger L')$ *with* $L \ddagger L'$ *being the family*

$$L \ddagger L'' = \{A \times \{i\} : i = 1 \ \& \ A \in L \text{ or } i = 2 \ \& \ A' \in L'\}.$$

4. Morphisms respecting processes

Morphisms defined above neither respect nor reflect processes in systems or paths in the corresponding graphs. The reason is very simple. The operations DOM and COD are not respected (reflected), that is the images (counterimages) of the corresponding sets are not equal but included only in their counterparts. So for a subset X_0 of the carrier of an L-graph $\mathbf{GL}= (X, \mathrm{dom}, \mathrm{cod}, L)$ and a morphism $f : \mathbf{GL} \to \mathbf{GL}$ of L-graphs it holds $f(\mathrm{DOM}(X_0)) \supseteq \mathrm{DOM}'(f(X_0))$ instead of $f(\mathrm{DOM}(X_0)) = \mathrm{DOM}'(f(X_0))$ and $f(\mathrm{COD}(X_0)) \supseteq \mathrm{COD}'(f(X_0))$ instead of $f(\mathrm{COD}(X_0)) = \mathrm{COD}'(f(X_0))$. In some special cases the image of a process in a concurrent system (path in the corresponding L-graph) can be included in a process (path) in the image of this system (L-graph) but this is rather an exemption than a rule. In order to obtain a notion of morphisms respecting (reflecting) processes (paths) one has to modify the above definitions replacing the inclusions by equalities and, if the limitation family (the system) is not safe, introducing a condition describing the way of transforming the limitation family or the corresponding set of configurations. Let $\mathbf{GL} = (G, L)$ and $\mathbf{GL}' = (G', L)$ be L-graphs. DOM, DOM', COD and COD' denote the operations of domain and codomain in the power graphs $\mathrm{Pow}(G)$ and $\mathrm{Pow}(G')$ respectively. The corresponding definitions can be of the form:

DEFINITION 4.1 (Strong morphisms of graphs with limitation). *A triple* $f : \mathbf{GL} \to \mathbf{GL}'$ *with* $f : L \to L'$ *being a function is called*:

(a) *a strong morphism of* **GL** *into* $\mathbf{GL}'$ (*L-morphism*) *iff f respects the limitation L, i.e. iff* $f(L) \subseteq L'$ *and*

$$f(\mathrm{DOM}(X_0)) = \mathrm{DOM}'(f(X_0)) \ \& \ f(\mathrm{COD}(X_0)) = \mathrm{COD}(f(X_0))$$

for every $X_0 \in L$.

(b) *a twisted strong morphism of* **GL** *into* $\mathbf{GL}'$(*TL-morphism*) *iff f reflects the limitation L, i.e. iff* $f^{-1}(L) \supseteq L$ *and*

$$f^{-1}(\mathrm{DOM}(X_0)) = \mathrm{DOM}(f^{-1}(X_0)) \ \& \ f^{-1}(\mathrm{COD}'(X_0)) = \mathrm{COD}(f^{-1}(X_0))$$

for every $X_0 \in L$.

The corresponding categories are subcategories of GLGRAPH, TGLGRAPH, SYS, and TSYS described above and the constructions of products and coproducts are the same.

5. Processes

By a *process-occurrence* in an L-graph $\mathbf{GL} = (X, \mathrm{dom}, \mathrm{cod}, L)$ we mean any path in the graph $\mathbf{G} = (L, \mathrm{DOM}, \mathrm{COD})$. The process-occurrences so defined correspond to a kind of images of a process seen by some observers. So a process-occurrence is simply a sequence of steps. A process can be seen as a set of process-occurrences. In the sequel we define processes as equivalence classes of a congruence of the free category generated by an L-graph, more precisely by the graph $(L, \mathrm{DOM}, \mathrm{COD})$ with L being the limitation family of the corresponding L graph **GL**. In the free category Paths(**GL**) generated by this graph one can define a limitation family $L_{\mathrm{Paths}}(GL)$ putting for a set $A \subseteq \mathrm{Paths}(\mathbf{GL})$

$$A \in L_{\mathrm{Paths}}(\mathbf{GL}) \Leftrightarrow \forall_{B,C \in \mathrm{Paths}(\mathbf{graph}(GL))} \forall_{\alpha \in \mathrm{Pow}(X)} A = B\alpha C \Rightarrow \alpha \in L.$$

PROPOSITION 5.1. *For every L-graph* $\mathbf{GL} = (X, \mathrm{dom}, \mathrm{cod}, L)$ *the family* $L_{\mathrm{Paths}}(GL)$ *is a limitation family of the graph* $\mathrm{Paths}(GL) = (\mathrm{Paths}(L), \mathrm{DOM}, \mathrm{COD})$.

Proof. We have to prove that for every $A \in L_{\mathrm{Paths}}(\mathbf{GL})$ it holds

$$\mathrm{DOM}(A) \in L_{\mathrm{Paths}}(GL) \ \& \ COD(A) \in L_{\mathrm{Paths}}(\mathbf{GL}).$$

So let $A \in L_{\mathrm{Paths}}(\mathbf{GL})$. We have

$$A = \mathrm{DOM}(A) \bullet A \ \& \ A \in L_{\mathrm{Paths}}(GL) \Rightarrow \mathrm{DOM}(A) \in L_{\mathrm{Paths}}(\mathbf{GL}).$$

The proof for the operation COD is similar.

PROPOSITION 5.2. *If the limitation family L of an L-graph* $\mathbf{GL} = (X, \mathrm{dom}, \mathrm{cod}, L)$ *is safe then so is the limitation family* $L_{\mathbf{Paths}}(GL)$ *as well.*

The family $L_{\mathbf{Paths}}(\mathbf{GL})$ defines not only a limitation of the parallel but also of the sequential composition of paths. If it is not safe, then one can find paths:

$$A = A_1(\alpha \cup \alpha_0) \quad \text{and} \quad B = (\beta \cup \beta_0)B_1$$

with $\mathrm{COD}(A) = \mathrm{DOM}(B)$ and $\alpha_0, \beta_0 \subseteq \mathrm{Vertices}(\mathbf{GL})$ such that

$$\mathrm{COD}(\alpha) = \beta_0 \ \& \ \mathrm{DOM}(\beta) = \alpha_0$$

which implies $A \cdot B = A_1(\alpha \cup \beta)B_1$. Now it can be $\alpha \cup \beta \notin L$, i.e. $A \cdot B$ is not an allowed path in **GL**. If the limitation family of the graph **GL** is safe, then the family $L_{\mathbf{Paths}}(\mathbf{GL})$ determines a subcategory path (**GL**) of the free category Paths(**GL**). It is freely generated by the L-graph **GL** in both categories GLGRAPH and TGLGRAPH.

The elements of the set Paths(L) described above are our candidates for a kind of "process presentations". A process in a system can be seen as a whole built up from some "elementary" processes by means of the operations of sequential and parallel composition. These elementary processes are just the steps modeled by the elements of the family L of independent sets of an L-graph.

EXAMPLE 5.1. Let us consider two objects, say P(roducer) and C(onsumer), performing cyclically the following activities:

1. getting from a given place, say Z, some materials
2. processing these materials
3. delivering the results of activity to the place z.

The producer works in the following way. He takes from the place Z (the arrow β_1) the materials mentioned above and some own products (perhaps money) from the place a (the arrow β_1) and put it to the place b. We want see this activity as an elementary indivisible process of P. Next, he put the results of his work to the place Z and materials to be performed to the place c. Now he performs his proper production-activity γ_1 put its results to a, and starts the whole cycles again. The consumer works analogously. We illustrate the system thus specified graphically as an L-graph as it is shown in fig. 1.

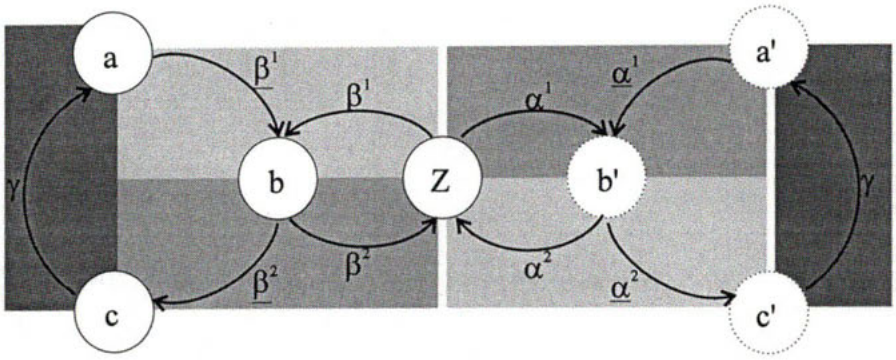

Figure 1: Producer - consumer system as an L-graph.

In this way of seeing this system one can distinguish six activities: (activities 1, 2, 3 of the producer and activities 1', 2', 3' of the consumer) and seven places in which the participating objects can be stored: (internal places a, b, c of the producer internal places of consumer a', b' c', and a place Z which is shared by producer and consumer). They are the minimal elements of a limitation family of this graph. Some other elements of this family, being elementary, sequential indivisible (but divisible in parallel) processes of this system are shown in fig. 2. below.

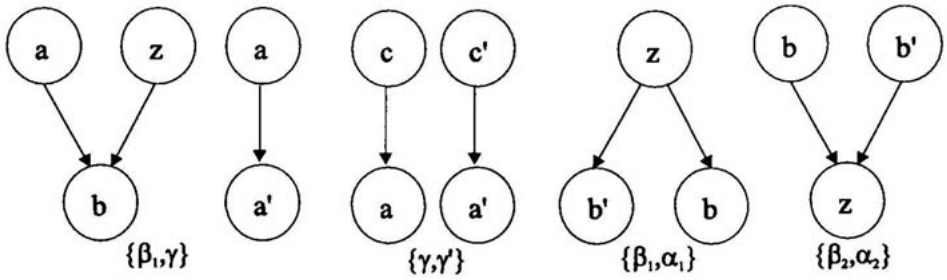

Figure 2: Elementary processes in the system from fig.1.

In the above example "elementary" means "belonging to the limitation family" of the L-graph being the model (in the sense described above) of considered system. It is evident that this meaning can be relativised to any L-graph; particularly to the graph $(\mathrm{Paths}(L), \mathrm{DOM}, \mathrm{COD})$ with L being a subgraph of the power graph $\mathrm{Pow}(G)$ of a graph G. In this case an "elementary process" can be seen as a part of another one which makes the using of the word "elementary" a bit problematic, but we use it in the above sense only. Now any path $A \in L_{\mathbf{Paths}}(\mathbf{GL})$, say A, can be "splitted" into a set

parts $(A) = \{A_1, A_2, \ldots, A_k\}$ in such a way that $A = \varphi(A_1, A_2, \ldots, A_k)$ with φ being a polynomial (an algebraic function) of a (partial) algebra of processes

$$\mathrm{Proc}(\Sigma) = (\mathrm{Paths}(L), \mathrm{DOM}, \mathrm{COD}, \bullet, +)$$

with

$$\bullet, + : \mathrm{Paths}(L) \times \mathrm{Paths}(L) -_{\circ} \to \mathrm{Paths}(L)$$

being (partial) operations of sequential and parallel compositions of processes. These operations can be defined in many ways. For the sequential composition the most natural domain seems to be the set

$$\{(\alpha, \beta) \in \mathrm{Paths}(L) \times \mathrm{Paths}(L) : \mathrm{COD}(\alpha) = \mathrm{DOM}(\beta)\}$$

i.e. the reduct $(\mathrm{Paths}(L), \mathrm{DOM}, \mathrm{COD}, \bullet)$ would be in this case the free category generated by the graph $(L, \mathrm{DOM}, \mathrm{COD})$. In the case of parallel composition is the question a bit more subtle, but the result of the parallel composition of processes, say p_1 and p_2 can be seen as a kind of "direct sum" of p_1 and p_2. Our problem is that we don't work with processes (such a notion hasn't be an defined yet) but with their *concrete presentations.* The simplest way to forget about presentation is to define a congruence of the corresponding algebra of processes and to define processes as its equivalence classes. So an algebra of processes in a system modeled by an L-graph GL can be seen as a quotient-algebra of an algebra of presentations of processes in the system. We describe it shortly as a quotient of the free category generated by the underling graph $(L, \mathrm{DOM}, \mathrm{COD})$ of **GL.** In order to define the corresponding relation (congruence) let us note that in the power algebra of a free category $\mathrm{Paths}(G)$ of all paths of a graph G one has a natural operation of an "addition"; the standard set union. Now let Q be the relation determined by the (weak) equation

$$A \bullet B \cup C \bullet D = (A \cup B) \bullet (B \cup D),$$

i.e. sets (of paths) of the form $A \bullet B \cup C \bullet D$ and $(A \cup B) \bullet (B \cup D)$ are identified. The relation Q is a (weak) congruence of every subcategory of the power algebra of $\mathrm{Paths}(G)$ for every graph G. Now for every L-graph $\mathbf{GL} = (X, \mathrm{dom}, \mathrm{cod}, L)$ we define the free L-category generated by **GL** as the quotient category $\mathrm{Paths}(L, DOM, COD)/Q$ equipped with the limitation, family

$$L(\mathbf{GL}) = \{[A]_Q : A \in L_{\mathbf{Paths}(G)}(\mathbf{GL})\}.$$

Having defined the free category with limitation one can define the refinement of concurrent systems as a substitution:

$$L \to L(\mathbf{GL}')$$

for L-graphs $\mathbf{GL} = (X, \mathrm{dom}, \mathrm{cod}, L)$ and $\mathbf{GL}' = (X', \mathrm{dom}', \mathrm{cod}', L')$ similar to the top-down programming style. Such substitutions may be not net morphisms in the "normal" sense (e.g. they can split processes into sets of parallel parts which make some problems with the consideration of a kind of time in a system) but they always respect (reflect) the reachability relation. Let us illustrate it by the example of the "producer-consumer" system described above.

EXAMPLE 5.4. We start from the terminal object of the category of the classical two sorted presentation of graphs (see fig. 3.).

Figure 3: The simplest graph - model of a system.

So our starting-point is a system consisting of one activity a and one situation z where $\mathrm{dom}(\alpha) = \mathrm{cod}(\alpha) = z$. The first step of our refinement is a refinement of the activity a into two activities α_1 and α_2. as it is pictured in fig. 4.

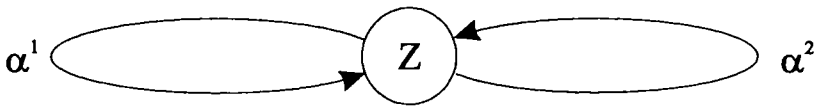

Figure 4: A refinement of arrows into parallel ones.
$f : a \mapsto a'$, $f : \alpha \mapsto \{\alpha_1, \alpha_2\}$

This refinement is a morphism of L-graphs because:

$$\begin{aligned}
&\mathrm{DOM}'(f(a)) \\
&\quad = DOM'(\{\alpha_1, \alpha_2\}) && \lceil \text{def. of the function } f \rceil \\
&\quad = z && \lceil a' \in S \Rightarrow DOM'(\{a'\}) = \{a'\} \rceil \\
&\quad = f(z) && \lceil \text{def. of the function } f \rceil \\
&\quad = f(\mathrm{DOM}(\alpha)) && \lceil a \in S \Rightarrow \mathrm{DOM}(\{a'\}) = \{a'\} \rceil
\end{aligned}$$

The proof for the operation COD is analogous. So the function f is a morphism of L-graphs. Now both α_1 and α_2 can be replaced by some processes; α_1 by a process $\alpha_{11}\alpha_{12}$ with an "internal" place b and α_2 by a process $\alpha_{21}\alpha_{22}$ with an "internal" place b' as it is shown in fig. 5. below:

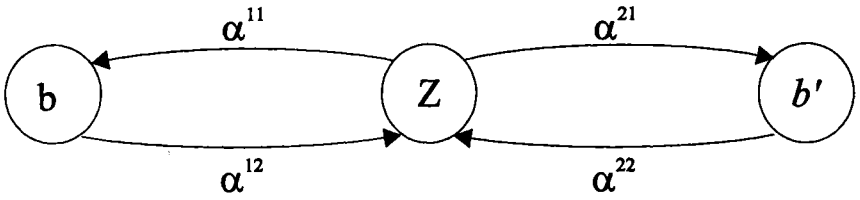

Figure 5: A refinement of arrows into paths (substitution of arrows by paths).
$z \mapsto z$, $\alpha_1 \mapsto \alpha_{11} b \alpha_{12}$, $\alpha_2 \rightarrow \alpha_{21} b' \alpha_{22}$.

It holds

$$\begin{aligned}
&\mathrm{DOM}'(f(\alpha_1)\\
&\quad = DOM'(\{\alpha_{11}, \alpha_{12}\}) && \lceil\text{def. of the function } f\rceil\\
&\quad = \mathrm{DOM}'(\alpha_{11}) && \lceil\mathrm{DOM}'(\{\alpha_{11}\alpha_{12}\}) = \mathrm{DOM}'(\alpha_{11})\rceil\\
&\quad = f\{a''\}\\
&\quad = f(a') && \lceil\text{def. of the function } f\rceil\\
&\quad = f(\mathrm{DOM}(\alpha_1))
\end{aligned}$$

Proofs for the operation COD and the remaining elements of this system are analogous. So we have proved that the function f is a morphism of L-graphs. The last L-graph can be "glued together" with some other L-graphs as it is shown in fig.6.. This gluing together consists of a kind of "synchronization"; some pairs (indicated by dotted boxes) of situations and activities are synchronized into situations and activities of a new system.

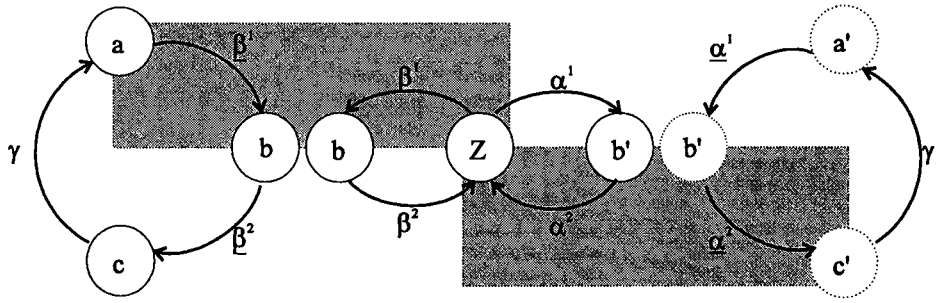

Figure 6: A synchronization of concurrent systems.

In the above picture the following elements of the L-graph representing the system under considerations are shown:

$$B = \{\underline{\beta_1}, \beta_1\}, B' = \{\underline{\alpha_1}, \alpha_1\}$$

This operation transforms the L-graph from fig 5.6. into the standard producer consumer system.

6. Concluding remarks

The presented formalism seems to be a good tool for a description of various problems concerning transformations of systems (refinement, product, coproduct etc.). There exist a lot of results in graph theory, its applications or related topics which can be easily used for examination of L-graphs. It seems one of such interesting connections between graphs and the formalism described in the paper can be found in an idea developed by Obtuowicz ([Obt94], [Obt96]). He proposes a kind of "hierarchy" on graphs which can be used as a tool for defining a large class of functions. Similar method can be useful for a description of some classes of relations or functions "generated by L-graphs". The possibility of a kind of "translation" of some operations on Petri nets into the language of L-graphs and vice versa will be showed in another paper. It allows for using them as a tool in resolving some typical problems occurring in net theory. The main difference between this approach and other ones - the possibility of modeling more general type of concurrence - constrain can be useful in the modeling of a generalization of concurrence understood as a binary relation.

References

[DMM91] P. Degano, J. Meseguer, U. Montanari, *Axiomatizing the Algebra of Net Computations and Processes*, Technical Rep. 1/91 of Comp. Sc. Dep. of University of Pisa.

[GLT] H. J. Genrich, K. Lautenbach, P. S. Thiagarajan, *Elements of General Net Theory*, in: Brauer W. (ed.) Net Theory and Applications, LNCS 84 Springer Verlag Berlin, Heidelberg, New York, 1980.

[Kor80] W. Korczyński, *An axiomatic characterization of an algebra of processes in concurrent systems*, ICS-PAS Report 400, Warsaw 1980.

[Kor88] W. Korczyński, *An algebraic characterization of concurrent systems*, Fund. Inform. 11, 1988.

[Kor96] W. Korczyński, *On a Notion of Concurrence*, Fund. Inform., 1, 1996.

[Kor?] W. Korczyński, *On a presentation of Petri nets*, to be published in Arch. Inform. Teoret. Stos.

[Ma77] A. Mazurkiewicz, *Concurrent program schemes and their verification*, Technical Report of Aarhus University, 1977.

[NP91] M. Novotny, Z. Pawlak, *Algebraic theory of independence in information systems*, Fund. Inform. 4, 1991.

[Obt94] A. Obtułowicz, *Graphical sketches, a finite presentation of infinite graphs*, manuscript, 1994 to be published in Fund. Inform.

[Obt96] A. Obtułowicz, *Diagrammatic presentation of computable functions, topological aspects of computational complexity*, manuscript, 1996.

[Pe73] C. A. Petri, *Concepts of Net Theory*, Proceedings of MFCS'73, Springer LNCS 1973.

[Win82] J. Winkowski, *An algebraic description of system behaviors*, Theoret. Comput. Sci. 21, (1982).

INSTITUTE OF MATHEMATICS
FACULTY OF MENAGEMENT
PEDAGOGICAL UNIVERSITY KIELCE
ul. M. Konopnickiej 21
25-406 KIELCE, POLAND

Received September 16, 1996.