Volume 13 Number 4 October 1989
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Informatica

A Journal of Computing and Informatics

Subscription Information

Informatica (YU ISSN 0350 – 5596) is published four times a year in Winter, Spring, Summer and Autumn (4 issues).

The subscription price for 1989 (Volume 13) is US$ 30 for companies and US$ 15 for individuals.

Claims for missing issues will be honoured free of charge within six months after the publication date of the issue.

Printed by Tiskarna Kresija, Ljubljana.

Informacija za naročnike

Informatica (YU ISSN 0350 – 5596) izide štirikrat na leto, in sicer v začetku januarja, aprila, julija in oktobra.

Letna naročnina v letu 1989 (letnik 13) znaša za podjetja 48000 din, za zasebne naročnike 12000 din, za študente 4000 din; posamezna številka 16000 din.

Številka žiro računa: 50101 – 678 – 51841.

Zahteva za izgubljeno številko časopisa se upošteva v roku šestih mesecev od izida in je brezplačna.

Tisk: Tiskarna Kresija, Ljubljana.

Na podlagi mnenja Republiškega komiteja za informiranje št. 23 – 85, z dne 29. 1. 1986, je časopis **Informatica** oproščen temeljnega davka od prometa proizvodov.

Pri financiranju časopisa **Informatica** *sodeluje Raziskovalna skupnost Slovenije.*

Volume 13 Number 4 October 1989
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Letnik 13 Številka 4 Oktober 1989 YU ISSN 0350 – 5596

Informatica

Časopis za računalništvo in informatiko

V S E B I N A

SYNTHESIS IN COMPLEX PROBLEM DOMAINS

INFORMATICA 4/89

Keywords: synthesis, complex systems, abstraction, hierarchical decomposition, structuring

Maksimilijan Gerkeš
Metalna Maribor

ABSTRACT:

Complex systems' synthesis becomes a bottleneck of production process. Some researchers denote this situation as crisis, others as "crisis". However, complex systems' synthesis especially those, which cannot be implemented within particular technology domain is an open problem.

The contribution gives a collection of procedures, developed with integration and some inovations of classical procedures and more advanced procedures. As a result, synthesis process can be based on abstraction, hierarchical decomposition, and structuring, while its derivation is nested in actual technological context, expressed through requests synthesized solution must satisfy.

POVZETEK: Sinteza v domeni kompleksnih problemov

Sinteza kompleksnih sistemov postaja vedno bolj ozko grlo produkcijskega procesa. Nekateri raziskovalci označujejo to stanje kot krizno, drugi kot "krizno". Kakorkoli, problem sinteze kompleksnih sistemov, posebej tistih, ki jih ni možno rešiti v domeni posamezne tehnologije, je dokaj odprt.

Prispevek podaja skupek postopkov izpeljanih, z integracijo in nekaj inovacijami, iz klasičnih postopkov in sodobnih postopkov sinteze. Kot rezultat lahko postopek sinteze gradimo na osnovi abstrakcije, hierarhične dekompozicije in strukturiranja, njegovo izvajanje pa je vgnezdeno v konkretni tehnološki kontekst izražen v obliki zahtev, ki jih mora izpolnjevati rešitev.

INTRODUCTION

Situation in high technology domains like CIM, flexible manufacturing, software, computer architectutes, VLSI, etc., seems similar regarding the requests for more efficient synthesis methods. For economic reasons synthesized solutions must be completely validated before manufacturing. A single error can cause exponential growth of operations to dispatch it. This calls for formal correct synthesis methods.

Complexity is common characteristic of systems in the above domains. It causes an enormous amount of operations, before the synthesis goal can be satisfied. Most of the contemporary synthesis methods seem to fail because of their too close technological orientation.

Rapid technology development disables efficient synthesis tools to be developed for each particular technology. Even if this is possible, there will still remain the problem of systems implemented in diverse technologies.

This clearly calls for synthesis methods, which will be technology independent and conceptualized on complexity and functionality issues of contemporary and advanced systems.

This contribution is based on system orientation to the synthesis problem, where current technology sets limits regarding system's functionality and structure. Such an approach allows that technology based descriptions are completely avoided until the solution representation level. Altough technology sets very exact limits during the synthesis process, technological objectives are not expressed explicitly, but reflect in

system structure and functionality.

An approach to synthesis is used, which is based on abstract system representation transformed through hierarchical decomposition and structuring to its counterpart, the solution. Abstraction and hierarchical decomposition have long been recognized as a means, which enables complexity reduction on one side and problem decomposition to several subproblems of lower complexity on the other side.

Structuring as a means to cope with complexity is more controversial and it seems there is no unisonous agreement about it.

However, combining abstraction and hierarchical decomposition in the synthesis process assures only that the initial and subsequent system structures are replicated and impressed in solution system structure. On the other hand it is well known that contemporary systems are supported with an enormous amount of software, which directs their actions and is a strict consequence of systems' structuring. To visualize this we can imagine that software through execution connects an object-flow graph that corresponds to problem structure and behaviour. This means that structuring is already extensively used in system design.

With respect to system represented with uncontrolled object-flow graph structuring requires three additional system structures. Its control structure directs subsystems , which can be at lowest representation level represented with uncontrolled object-flow graphs, connected in space and time. Particular strategy of control used is of secondary importance. Its memory structure assures object validity during controlled execution. Transfer structure delivers objects to subsystems' inputs. Both memory and transfer structures are under control of control structure. Those structures are usually partitioned throughout the system following the principles of distribution and hierarchy. More intensive structuring results in complex control scheme, which is usually but not necessary expressed in a form of control code or software. Making software more close to human comprehension capabilities does not change its nature. However, it is interesting that structuring does not necessary impose any control code or software.

This point of view allows that control code or software are determined as structuring side product. This is a conclusion, which sounds quite heretic, however, it gives at least theoretical possibility for automatic software development.

Basic structuring concept can be explained starting with a system with uncontrolled flow of objects. Observe that not all resources of such a system are active simultaneously. Assume a system in which resources are disconnected and a control mechanism, which is capable to connect them in time and space. Under certain conditions behaviour of both systems can be identical.

This simplified description, structuring is based on, should be completed with an observation that identical system resources can be shared when appropriate. To take advantage of this possibility system structure have to be completed with memory and transfer structures.

If the behaviour of a system have to change, its control functions which determine resource connections will be appropriately modified. The simplest way to do this is through program memory, which is a part of system's control structure.

Even when we look to software as problem description, system should be capable of its recognition and physical system restoration to solve the problem. A possible way to do this is that problem is represented with problems solvable with physical system resources. However, this introduce structure that have to be implemented with physical system in isomorphic way, directly or with appropriate time and space partition.

System synthesis can now be conceptualized based on the problem expressed in a form of abstract system, which have to be transformed through hierarchical decomposition and structuring to a solution, a system expressed with resources and structure that can be implemented in physical world.

Problem solvability seems to be in tight connection with its representation. Different representations can be used to express particular view on the problem. They are equivalent in the sense that they express the same behaviour in different ways. Synthesis procedures proposed are defined for each representation. Change of one representation to another is orderly developed.

1. OVERVIEW OF THE SYNTHESIS PROCESS

Problem specification consists of two parts. Problem definition part determines an abstract system for which the synthesis process have to determine a solution expressed in the form of physical system specification to be implemented in predetermined technology domain. Behaviour of physical system can be recognized as behaviour of abstract system merely through the use of suitable abstraction, otherwise no relation exists between the systems.

The rest of problem specification sets requests for the solution. Two classes, called the solution classes are determined based on the requests. Resource class consists of resource specifications, physical system should be built of. Resources determined with the resource class are generally composed and can be represented with resource structures of known behaviour. This usually allows resource class structuring through suitable relations. Abstraction through equivalence relation regarding resource behaviour reduces representation complexity and allows unnecessary details to be

neglected at earlier synthesis steps. More sophisticated abstraction procedures to reduce representational complexity are described in section 3.

Structure class consists of abstract resource structures. Physical system structure and its abstractions should be isomorphic to structures build of structures from structure class. Structure class is partially ordered with regard to substructure relation. Its representation complexity can be reduced through structure abstraction. Structure class can be empty. From the synthesis point of view this means that no structure requirements are set regarding the solution.

Intuitively, abstract and physical systems can be delimited with the notion of distance, which is a measure proportional to the number of synthesis steps required to reach the solution. Its numeric determination provides estimate about problem complexity - P, NP complexity as the roughest measure, for example. An argument will be given to show that the synthesis problem belongs to the domain of NP - complete problems, in general.

Initial synthesis steps must assure the match *between problem and an abstract solution derived from* solution classes. The match is found through structures' isomorphism and identical behaviour of corresponding resources. Further problem abstraction and structuring may be needed to assure it in explicit or implicit way. It is assumed, that some generic knowledge about problem solvability in the context of both solution classes is available. In the opposite case problem can be solved with an exhaustive search only. This is probably the strongest reason why a kind of systematic frame for the synthesis process is needed.

Synthesis procedures cannot be deterministic because of partial ordering of structure class. Structure limitation can play a dominant role in limiting the number of synthesis steps. The other limiting factor for the synthesis step count can be searched for in resource class. Resource class consisting of resources with simple behaviour characteristics will necessary complicate the synthesis. With proper problem structuring and composite resources built of resources from resource class this problem can be reduced. However, the effect of this depends on knowledge about problem characteristic properties.

Synthesis process can be stated as a combination of hierarchical problem decomposition and structuring. Hierarchical problem decomposition is a process basically opposite to abstraction. In our case it consists of refinement and interpretation processes. With its application problem is stepwise decomposed to a structure consisting of mutually dependent subproblems each of them have lower complexity with respect to the initial problem and subsequent subproblems obtained at earlier decomposition steps. This increases problem structure complexity and causes a consequence that structures of higher problem representation levels are replicated at lower representation levels. Applying structuring to arbitrary intermediate representation level results in structure change, while preserves representation level and behaviour.

Since hierarchical decomposition preserves structure and structuring preserves representation level it is clear that requirements expressed through solution classes can be met only if both hierarchical decomposition and structuring are applied during the synthesis process.

Synthesis process can be recognized as partial ordering relation between problem, intermediate solutions, and solution, where the solution or solutions are those intermediate solutions, which satisfy requests expressed through both solution classes.

To find a path between problem and solution and to avoid searching over the whole partially ordered structure of possible intermediate solutions, decisions based on their properties can drive the synthesis process. A minimal condition for an intermediate solution to be accepted for further synthesis is that it can be *expressed with the objects of both solution classes.* Different synthesis strategies can be used to determine the synthesis process. The simplest one and probably the less useful is one, where hierarchical decomposition to the resource class representation level is done first. This intermediate solution is then structured to assure compatibility with objects from structure class. However, structuring can become a task of excessive complexity within this approach. To avoid such situations, hierarchical decomposition and structuring are combined during the synthesis process. This assures manageable subproblems' complexity.

Applying structuring and abstraction to both solution classes match between intermediate solution and structure consisted of objects from both solution classes can be found at each intermediate representation level. When such a match can be found between an intermediate solution and a structure consisting of objects from both solution classes without any abstraction such intermediate solution is accepted as a solution. Structures' isomorphism and identical behaviour of corresponding resources assure systems' equivalence.

2. REPRESENTATIONS

Different views can be applied to represent the same system. In conventional one system is represented as a structure of interconnected resources with known behaviour. For system behaviour representation it is interesting to represent resource inputs with input positions and its outputs with output positions. Positions represent perception points from which object flow is oriented toward the resource or from it. For

two connected resources corresponding position plays double role. It is output position for one resource and input position for the other. Labelled bipartite directed graphs are suitable abstraction for this view on system's structure. System's behaviour can be represented with the behaviour of system resources. Based on resource positions collection of objects flowing to and from resource can be determined. Each collection corresponding to particular resource has objects represented with $m+n$ tuples corresponding to objects on input and output resource positions. Such a collection can naturally be represented with function or relation, depending whether resource reacts to the same input in deterministic or nondeterministic way. Relation between functional and relational resource behaviour will be clarified later in this section.

During the synthesis process more attention can be given to system structure since its behaviour remains the same throughout this process. For this reason resource structure can be represented neglecting positions in system structure.

Definition: Resource structure is labelled directed graph $G = (V,E)$, where $V = \{v_1,\ldots,v_n\}$ is a set of resources, and $E = \{e_1,\ldots,e_n\} \subseteq V \times V$ is a set of resource connections.

Resource labelling is defined with functions which assign labels to resources and connections and enable system behaviour recognition.

Returning to the initial system model represented with bipartite directed graph it can easy be recognized that this representation is in close connection with data-flow graphs defined in [1] , [2] . In our case modification of this definition will be used for representing system's structure and behaviour.

Definition: Controlled object-flow graph is labelled bipartite directed graph $G = (AULUK, E)$, $L \cap K = \emptyset$, $A = \{a_1,\ldots,a_n\}$ is a set of action nodes, and $LUK = \{l_1,\ldots,l_m, k_1,\ldots,k_r\}$ is a set of links, where $K = \{k_1,\ldots,k_r\}$ is a set of control links.
$E \subseteq (A \times (LUK) \cup (LUK) \times A)$ is a set of branches defined so, that the following restrictions are satisfied,

$$(a_i, l_k) \in E \,\&\, (a_j, l_k) \in E \Rightarrow a_i = a_j, \quad l_k \in LUK,$$

$$(l_k, a_i) \in E \,\&\, (l_k, a_j) \in E \Rightarrow a_i = a_j, \quad l_k \in LUK,$$

$1 \leq i, j \leq m,\ 1 \leq k \leq m + r.$

Two kinds of behaviour can be assigned to action nodes of controlled object-flow graph. One is represented with function, the other with controlled function. In the final case action node must have at least one input control link.

Let $f: X_1 \times \ldots \times X_n \longrightarrow Y$ be a function defined on known sets $X_1,\ldots,X_n$, Y. Controlled function g for function f is a function,

$$g: \text{Bool} \times X_1 \times \ldots \times X_n \longrightarrow Y, \quad \text{Bool} = \{0,1\}$$

$$g(p, x_1,\ldots, x_n) = \begin{cases} f(x_1,\ldots x_n) & ; \ p = 1 \\ \text{undef} & ; \ p = 0. \end{cases}$$

Figure 2.1 shows controlled object-flow graphs for functions f and g.

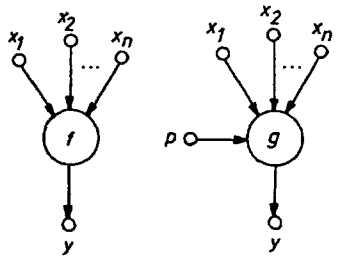


Figure 2.1: Controlled object-flow graphs for functions f and g

To avoid confusion instead of $g(0, x_1,\ldots, x_n) = \text{undef}$ we can define $g(0, x_1,\ldots,x_n) = z$, $z \in Y \cup \{z\}$. Object z has different interpretations in different implementation situations. In electronic systems it can be used to represent a high impedance condition, for example. In general, it represents no object or no activity situations.

The notion of controlled function can be extended to more sophisticated cases of controlled execution. Examples can be found in section 5.2. One further example is provided below,

$$g: A \times X_1 \times \ldots \times X_n \longrightarrow Y, \quad A = \{a_1, \ldots, a_m\}$$

$$g(a, x_1, \ldots, x_n) = \begin{cases} f_1(x_1, \ldots, x_n); a = a_1 \\ \ldots \\ f_m(x_1, \ldots, x_n); a = a_m. \end{cases}$$

However, it can be shown that they are expressible within the initial definition of controlled function. The notion of controlled function allows formal connection between controlled object-flow graph and control-flow graph.

Definition: Control flow graph is labelled bipartite directed graph $G = (QUA, E)$, where $Q = \{q_1, \ldots, q_n\}$ is a set of states, and $A = \{a_1, \ldots, a_m\}$ is a set of function nodes. $E \subseteq ((Q \times A) \cup (A \times Q))$ is a set of branches.

System's behaviour expressed through control flow graph is based on state concept. Particular state, when recognized active, fires corresponding functions. This situation is effectively modelled with a pair, consisting of decision function and controlled function, which is fired with decision function's value. Section 5 gives more detail about this topic.

To allow change of representation during the synthesis process conversions between the above representations are defined. To increase synthesis flexibility they are of one - to - many type.

Conversions between resource structure and controlled object - flow graph are based on graph isomorphism with deleting or inserting link nodes. However, those conversions are usually applied with

structuring, what makes correspondence between structures less explicit.

Conversions between controlled object - flow graph and control - flow graph are less evident. Figure 2.2 gives an example of controlled object - flow graph to control - flow graph conversion. Formal basis for those conversions is given in section 5.

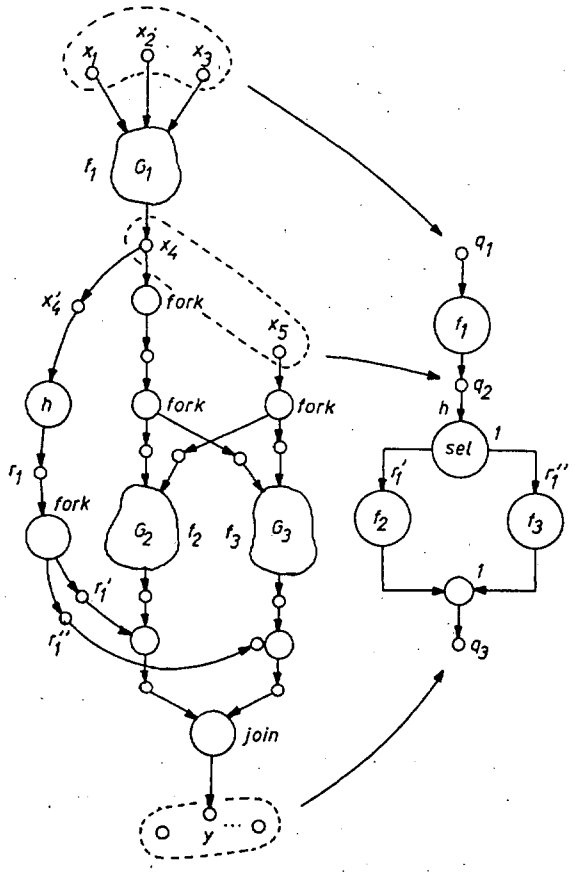


Figure 2.2: Example of controlled object - flow graph to control - flow graph conversion

Nodes of control - flow graph marked with $\perp$ are inserted to increase readability. From the formal point of view they are not necessary.

To describe system's behaviour production rules expressed with if then clauses can be used. We will show that such system descriptions can be converted to controlled object - flow graph descriptions and vice versa.

Originally conditional statements are used to represent if then clauses.

Take $R(x_1, \ldots, x_n) \longrightarrow Q(y_1, \ldots, y_m)$ as an example. Define characteristic functions for R and Q,

$$p = h_R(x_1, \ldots, x_n) = \begin{cases} 1 ; & R(x_1, \ldots, x_n) \\ 0 ; & \overline{R(x_1, \ldots, x_n)} \end{cases}$$

$$r = h_Q(y_1, \ldots, y_m) = \begin{cases} 1 ; & Q(y_1, \ldots, y_m) \\ 0 ; & \overline{Q(y_1, \ldots, y_m)}. \end{cases}$$

Implement h_Q with controlled function g_Q,

$$q = g_Q(p, y_1, \ldots, y_m) = \begin{cases} h_Q(y_1, \ldots, y_m); & p = 1 \\ 1 ; & p = 0. \end{cases}$$

It can easy be verified that p, r, and q determine truth table for conditional. Corresponding controlled object - flow graph is shown on figure 2.3.

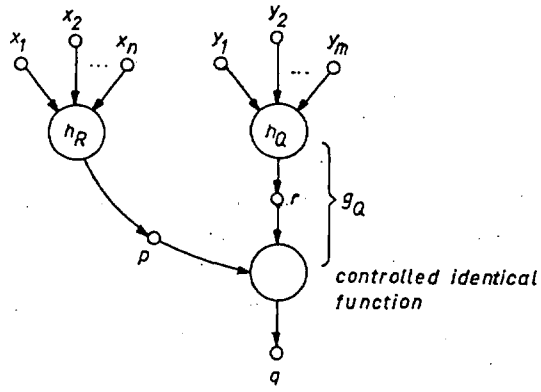


Figure 2.3: Controlled object - flow graph representing conditional clause

This representation allows inference processes' modelling in functional way.

Model of condition - action rule can be developed with slight modification of the above construction.

Let $R(x_1, \ldots, x_n) \longrightarrow Q(x_1, \ldots, x_n, y)$, and $Q(x_1, \ldots, x_n, y) \iff (y = f(x_1, \ldots, x_n))$.

Define characteristic function h for R, which defines domain of f.

$$p = h(x_1, \ldots, x_n) = \begin{cases} 1 ; & R(x_1, \ldots, x_n) \\ 0 ; & \overline{R(x_1, \ldots, x_n)}, \end{cases}$$

and implement f with controlled function g,

$$g(p, x_1, \ldots, x_n) = \begin{cases} f(x_1, \ldots, x_n) ; & p = 1 \\ \text{undef} ; & p = 0. \end{cases}$$

Controlled object - flow graph that corresponds to the above construction is shown on figure 2.4.

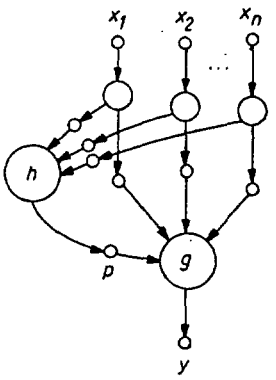


Figure 2.4: Controlled object - flow graph representing condition - action rule

It is interesting to note that observing behaviour only on input and output links of a graph we are not able to identify whether function f is implemented directly or with function g.

When dealing with systems it is usually presupposed that resources behave functionally. For systems described with relational connectives the above assumption may not be true. A minor modification allows that resources, which express relational behaviour can be treated in functional or relational way depending on point of view used. Since the complete treatment for arbitrary relations is relatively extensive, we will limit this presentation to binary relation R(x,y), which is partially closed with object a, R(a,y),

R(x,y)

⋮ ⋮
a b_i
a b_{i+1}
...
a b_{i+j}
...
a b_{i+n}
⋮ ⋮

Applying object a to a resource, which behaves corresponding to R, its response will be nondeterministic since it can delivers any object from b_i, b_{i+1}, ... ,b_{i+n} to its output position to satisfy R.

The behaviour of a such resource can be represented in functional way, if resource response is determined with a sequence function replacing R. Sequence function determines the order in which resource reacts with output objects to the same input determined with object a. Figure 2.5 gives tabular definition of sequence function and corresponding controlled object - flow graph for this case.

f:

x^k	y^{k-1}	y^k
a	b_i	b_{i+1}
a	b_{i+1}	b_{i+2}
...		...
a	b_{i+n}	b_i

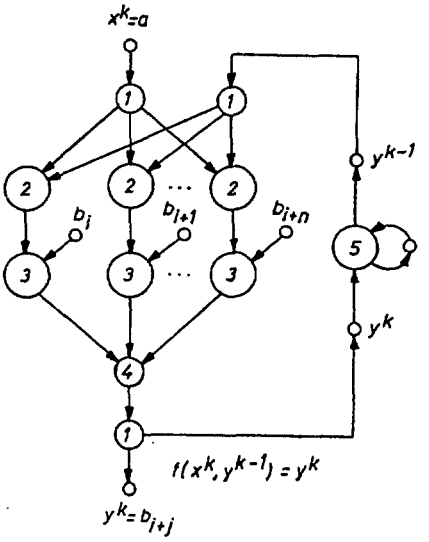


Figure 2.5: Tabular definition of sequence function and corresponding controlled object - flow graph

Identifiers 1 through 5 stand to identify functions fork, decision Boolean functions, controlled identical functions, function join, and memory function. Functions fork replicate input objects, decision Boolean functions control corresponding identical functions to deliver particular object b_{i+j} to function join, which is a treshold function and delivers object b_{i+j} to output link. Memory function assures necessary delay. Assumption is made that y^{k-1} has allways a value from $b_i,\ldots,b_{i+n}$. Two notes are necessary about the above presentation. First, it is simplified to serve conception presentation only, and second, sequence function can be defined in a number of different ways.

3. REFINEMENT AND INTERPRETATION

Using abstraction representation complexity can be reduced to a manageable level. During the synthesis process the situation is reversed, complexity grows, since this process is basically opposite to abstraction.

Assume a solution system represented with resource structure and resource behaviour. Using a substructure relation to determine an arbitrary substructure observe that its behaviour can be described with collection of objects which correspond to its input and

output positions. This enables that substructure is replaced with a resource having input and output positions that correspond to substructure input and output positions. The replacement causes lower system structure complexity.

Representing the system in each possible way with replacing substructures with resources while preserving their input output behaviour results in a class of systems with different structures and the same input output behaviour.

An important hypothesis can be made at this point. System structure abstraction has sense only if the substructure behaviour can be expressed with its input output behaviour[1]. Represent collections of objects which determine resources' behaviour with tables and assume they are of finite leught. Resource behaviour is represented with single table, while resource substructure behaviour is represented with a structure of interdependent tables. Select a pair resource, resource substructure with identical behaviour. The question arises whether the structure of interdependent tables can be developed based on the information obtained from single table.

Results of empirical study show that this is possible. However, for a formal proof of the above claim additional research is needed.

Results of mentioned study show that refinement based on the above concept is generally NP - complete problem since lower complexity bound grows exponentially with the number of table entries in non - trivial cases. Refinement is nondeterministic process that can be automated. Automatic algorithm development is possible. Controlled functions are necessary to obtain all possible refined versions of particular function. Refinement can be done on uncompletely specified tables. If this causes a lose of significant information, obtained algorithm will not be optimal.

Example 3.1 gives tabular refinement for selected case of binary addition and corresponding controlled object - flow graphs, which have no control links for this case. Only one path of the whole refinement process is presented.

Example 3.1

T

f:

a_1	b_1	a_o	b_o	c_2	d_1	d_o
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	1	0	0	1	0	0
1	1	0	1	1	0	1
1	0	1	0	0	1	1
1	0	1	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

G:

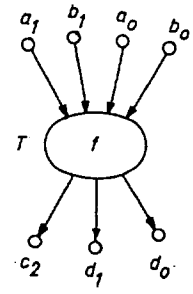
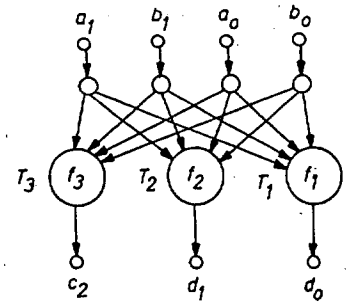


Table T represents binary addition decompose to tables T_1, T_2, and T_3, what results in the following controlled object - flow graph.



T_1

f_1:

a_1	b_1	a_o	b_o	d_o	
0	0	0	0	0	1.step
0	0	0	1	1	
0	1	0	0	0	2.step
0	1	0	1	1	
0	0	1	0	1	
0	0	1	1	0	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	1	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

T_{11}

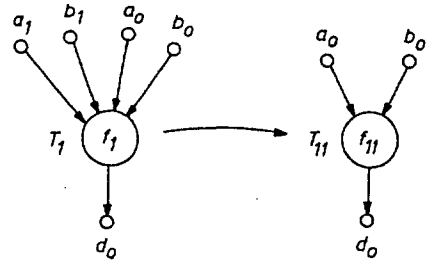
f_{11}:

a_o	b_o	d_o
0	0	0
0	1	1
1	0	1
1	1	0

T_1 becomes T_{11} after deleting redundant entries.

[1] Loops and circles in resource substructure can introduce additional substructure inputs and outputs which are local to it.

Corresponding controlled object - flow subgraph is reduced as represented below.



f_2:

a_0	b_0	a_1	b_1	d_1
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

f_{21}:

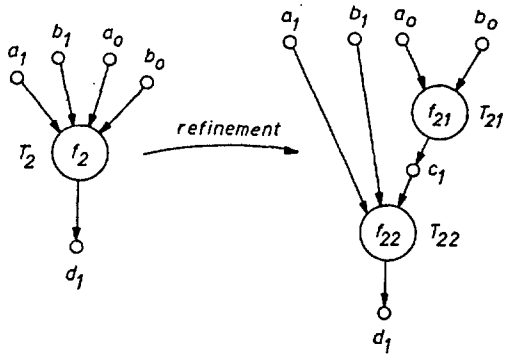
a_0	b_0	c_1
0	0	0
0	1	0
1	0	0
1	1	1

f_{22}:

a_1	b_1	c_1	d_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

T_2 is decomposed to T_{21} and T_{22}.

Corresponding controlled object - flow subgraph is refined as represented below.



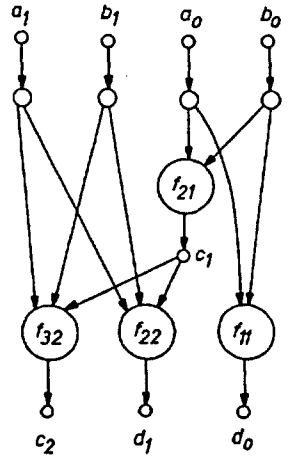
Refinement of T_3 is analogous to the refinement of T_2. The result is,

$T_{31} = T_{21}$,

f_{32}:

a_1	b_1	c_1	c_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Controlled object - flow graph obtained after refinement of T_1, T_2, and T_3 is given below.



Because of close relation between tabular and expressional function representation, the former can be thought of as tabular structure abstraction. Refinement procedures are generally developed for expressional function representation. Since this type of refinement is widely known its presentation will be avoided.

3.1 Function refinement

This subsection states conditions that have to be satisfied with function refinement. Figure 3.1 shows graph of relations and controlled object - flow graphs for functions f and g, which is composition of functions $g_1, \ldots, g_r$ obtained through refinement. Refined function g and initial function f have to satisfy the relation $f = h_1^{-1} \circ g \circ h_2 = g$, and h_1, h_2 are identical functions.

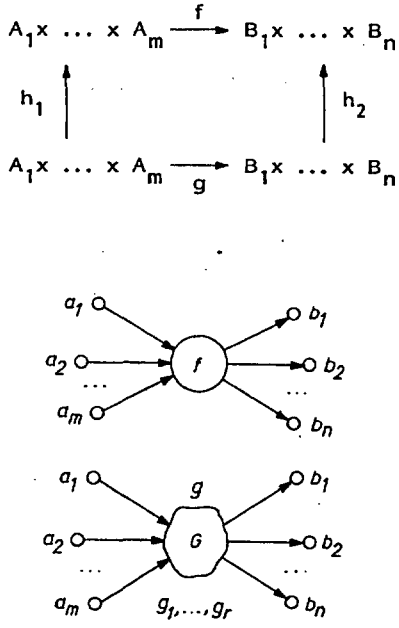


Figure 3.1: Function refinement

Dependent of actual representation defined in section 2 function refinement is completed with corresponding graph refinement. Procedures for them can be found in [2] , [3] .

Refinement can be thought of as a case of interpretation. However, we will give only basic definition and neglect this possibility.

Figure 3.2 shows graphs of relations and corresponding controlled object - flow graphs when interpreting function f with function g.

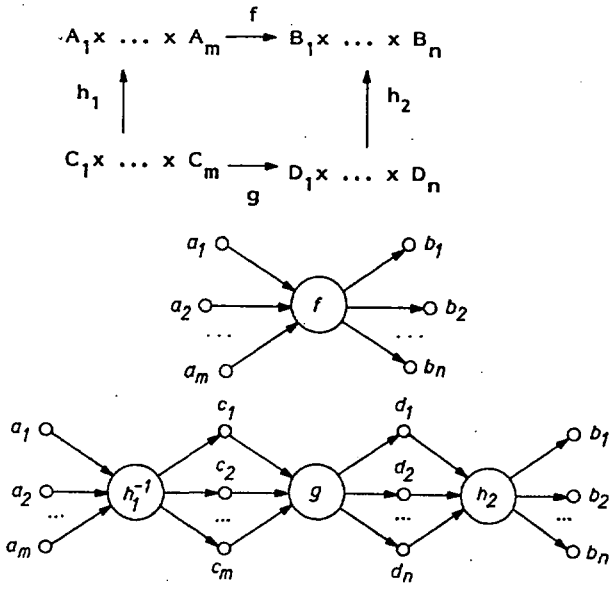


Figure 3.2: Interpretation of f with g

When interpreting function f with function g the relation $f = h_1^{-1} \circ g \circ h_2$, where h_1 and h_2 are surjective and possibly partial functions must be satisfied. Functions h_1 and h_2 assure compatibility with the environment. They can be stepwise removed with interpretation of environmental functions.

Since h_1^{-1} is generally relation this can cause formal inconsistency with controlled object - flow graph behaviour. Since h_1^{-1} can be represented with sequence function as illustrated in section 2 this can cause no serious problems. On the other hand h_1^{-1} and h_2 can be stepwise removed as mentioned above. The inconvenience can be avoided when h_1 is bijective.

3.2 Object interpretation and refinement

Until now objects were considered as integral units. However, for the reason of efficiency such observation is too restrictive. To avoid this, object representation can be adapted to problem representation level.

Intuitively, an arbitrary object composed of several objects can be viewed as integral unit or as a structure of objects positionally[2] determined. In first case the fact that object is composed is neglected. Several objects must satisfy some known relation which determine their mutual positions to be recognized as integral unit . To represent such structures n - tuples are used.

Object refinement and interpretation[3] are defined in connection with corresponding system's functions. Figure 3.3 shows a graph of relations and corresponding controlled object - flow graphs when refining objects of function f with objects of function g.

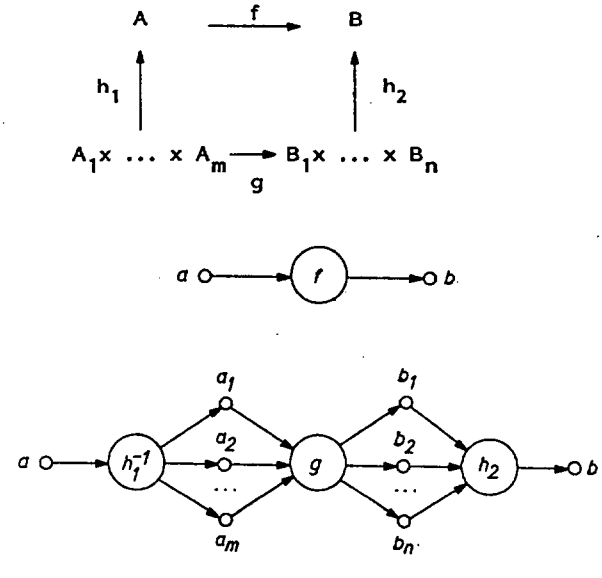


Figure 3.3: Object refinement and interpretation

[2] The notion of position is not restricted to physical position only.

[3] Difference between refinement and interpretation is semantical. Refinement is restricted to common semantic domain, while interpretation is not.

Object refinement and interpretation must satisfy, $f = h_1^{-1} \circ g \circ h_2$, where h_1 and h_2 are surjective and possibly partial functions. With further refinement functions h_1 and h_2 can be stepwise removed from the system.

Object refinement and interpretation can be extended to objects from $A_1, \ldots, A_m, B_1, \ldots, B_n$. Stepwise refinement of an object results in a tree structure, an example of which is shown on figure 3.4.

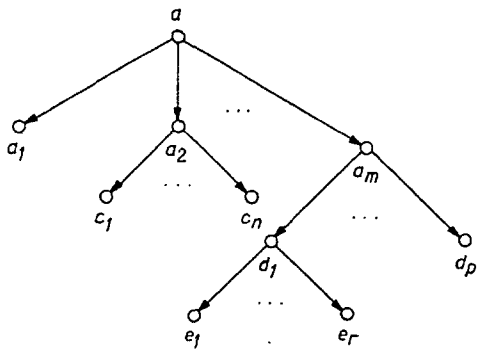


Figure 3.4: An example of object representation hierarchy

Refinement and interpretation have an interesting property, they preserve structure of previous higher represantation levels.

For illustration and application of this property assume an abstract system S_1 and a physical system S_2 and imagine both systems' behaviour in state spaces. Select an arbitrary state of S_1 and with interpretation and refinement determine corresponding state of S_2. This state will cause in S_2 a sequence of state changes. Assume a state from the sequence that has a corresponding state in S_1 and assume that this state is next state of initially selected state of S_1. The impression an observer seeing only the states of S_1 will have, is that S_1 changes states altough in reality state changes of S_1 are consequence of state changes in system S_2. Figure 3.5 illustrates the above explanation.

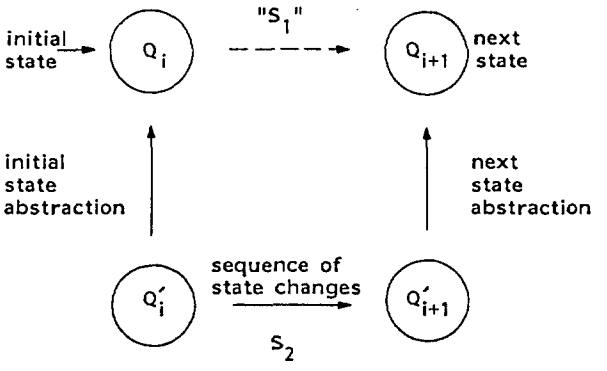


Figure 3.5: Implementation of S_1 with S_2

Refinement and interpretation were defined for all representations given in section 2. More details about them can be found in [2] and [3] .

4. STRUCTURING

Assume arbitrary system represented with resource structure. Select a resource and denote it with r. Since system representation level can vary such resource can represent resource substructure of lower representation level.

Determine those positions of system resource structure which carry objects from system input positions to resource r input positions. In the resource structure positions determine resource substructure which input positions are part of system resource structure input positions, while its output positions comprise resource r input positions.

For a subsystem, which belongs to the resource substructure determine composed relational expression denoted with E, which is satisfied when objects' state on resource r input positions enable resource activation. Instead with resource r function f determine its behaviour with relation corresponding to f and denote corresponding expression with Q. Form conditional $E \rightarrow Q$, which is satisfied after resource r delivers objects on its output positions. In the context of structuring it will be called control condition, since it allows uniform determination of each subsystem's state which causes resource r action.

Control condition can be rewritten as $E \rightarrow R$ & $R \rightarrow Q$, where R denotes relational expression describing states on resource r input positions, which cause its action. This form of control expression enables associative approach to control.

Determine characteristic function f_E for E,

$$f_E : A \longrightarrow \text{Bool}$$

$$f_E(a) = \begin{cases} 1 ; & E \\ 0 ; & \bar{E}, \end{cases}$$

where A stands for subsystem's states and Bool determines {0, 1} .

Characteristic function f_E will be interpreted as decision function since its value decides about resource r activation. Implement resource r function f with controlled function g as described in section 2. Join functions f_E and g into a pair. Collection of such pairs determined for each system's resource enables system behaviour and its structure reconstruction.

Figure 4.1 illustrates a distinction between the initial system and the system determined with collection of pairs decision function, controlled function.

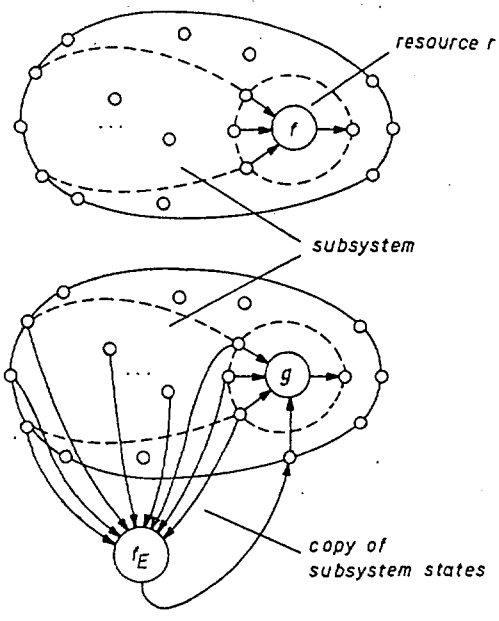


Figure 4.1: Distinction between systems

Define in A equivalence relation such that all states which activate the same subsystem's resource and resource r belong to the same equivalence class. Actually, more than one resource can correspond to particular equivalence class. Denote the set of equivalence classes with A/R and define bijective function from A/R to Q, where Q stands for the set of states. Based on each state q from Q it can be uniformly determined which resource(s) have to be activated as a consequence of this state, because bijective connection between A/R and Q.

Assume q from Q corresponds to equivalence class of states which causes resource r activation. Decision function corresponding to r can now be defined on set Q. To identify, when state q is active, what means that corresponding subsystem is in state which activates resource r, state transition function $f_s : Q \rightarrow Q$ is defined. It imitates subsystem state transitions from initial state to state corresponding to state q.

This model of control can now be expressed with the relations,

$$f_s(s)=q, \quad f_E(q)=1, \quad g(1, x_1, \ldots, x_m) = f(x_1, \ldots, x_m),$$

where s is the state preceeding state q, f_E is decision function, and g is controlled function corresponding to resource r function f.

Process of control as described above, can be defined for arbitrary system resources. There are no restrictions to limit it to a single state transition function. Extension of the model that state transition function enables n-way branching including loop control is relatively simple and will not be considered here. State approach to control is not the only one that can be developed based on decision function. In fact, as far as recognized all popular control strategies and combinations of them can be developed on that base.

One can easy recognize that if parts of control conditions are not necessary strictly distinct. This allows their logic composition, what results in decision function of the form,

$$f_E : A \rightarrow Bool^k, \quad k > 1.$$

More than simple resource can be controlled with a decision function and consequently more resources can be controlled with state transition function as mentioned earlier.

A pair decision function, controlled function enables structuring. A set of such pairs which determines the system can be partitioned to disjunctive subsets. The same effect can result from system partition on the set of disjunctive subsystems.

System integration based on its arbitrary partition can be realized in more different ways. Before do this we have to develop object transfer functions and memory functions. Transfer functions realize object transfer between positions, while memory functions assure that objects retain particular positions so long as determined with control structure.

4.1 Transfer functions

Flow of objects between two arbitrary subsystems separated with a partition can be implemented with selecting and distributive functions.

Assume a subsystem which has to be connected to another subsystem over positions $p_1, \ldots, p_n$. Denote sets of objects corresponding to positions with $X_1, \ldots, X_n$, and let $X = X_1 \cup \ldots \cup X_n$.

Selecting function Π is defined as,

$$\Pi : N \times X_1 \times \ldots \times X_n \rightarrow X$$

$$\Pi(i, x_1, \ldots, x_n) = \begin{cases} x_i, & 1 \leqslant i \leqslant n \\ \text{undef}, & \text{otherwise.} \end{cases}$$

The set of natural numbers N above can be replaced with arbitrary linearly ordered set. Indexes of $x_1, \ldots, x_n$ become objects of such set. Linear ordering can be avoided if selecting function is defined in tabular form. Both approaches to define selecting operation can be mixed. Levels of indirection can be built to the above definition to determine particular position of n-tuple.

Figure 4.2 shows controlled object - flow graph of selecting function.

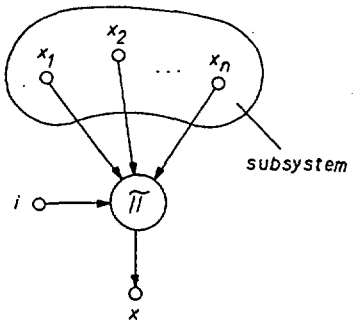


Figure 4.2: Controlled object - flow graph of selecting operation

Distributive function delivers objects obtained from selecting function to prespecified positions of a subsystem.

Distributive function is defined as,

$$\delta : N \times X \longrightarrow Y_1 \times \ldots \times Y_m$$

$$(y_1, \ldots, y_m) = \delta(j,x) = \begin{cases} (\text{undef}, \ldots, x, \ldots, \text{undef})^4, y_j = x, \\ \qquad\qquad 1 \leq j \leq m \\ \text{undef}, \quad \text{otherwise.} \end{cases}$$

Variables $y_1, \ldots, y_m$ correspond to subsystem input positions. Pair (i,j) determines a path from output position of one subsystem to input position of another subsystem. Composition of Π and δ can then realize arbitrary connenction between subsystems.

Figure 4.3 shows controlled object - flow graph of distributive function.

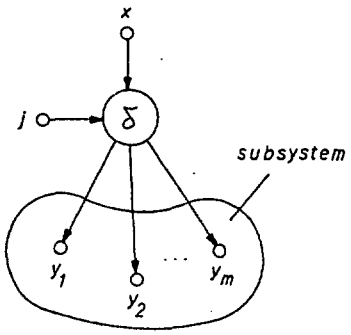


Figure 4.3: Controlled object - flow graph of distributive function

Selecting and distributive functions can be composed to realize arbitrary complex networks, which are capable of transfering specified amount of objects in time and space.

More requests for transfer than transfer paths available can exist simultaneously. Such requests are paid with appropriate time - space partition of transfer.

Selecting and distributive functions are not the only transfer functions possible. However, they are sufficient to implement arbitrary connection between two subsystems. They can be decomposed and expressed with controlled function defined in section 2.

4.2 Object representation in recursive domain

Let X be a set of objects to be represented in recursive domain. This representation is achived with function,

$$f : N \times X \longrightarrow X$$

$$f(i, x) = y$$

Pair (i, x) is represented with the notation x^i.

$$f(x^i) = \begin{cases} x, & n_1 \leq i \leq n_2, \quad n_1, \ n_2 \text{ from N} \\ \text{undef}, & \text{otherwise.} \end{cases}$$

The value of x is represented in domain which is linearly ordered set. Because of simple notation the set of natural numbers was selected, otherwise a set of states can be used, since it is linearly ordered with the state transition function.

In the real systems objects may have the ability to retain particular position - objects in mechanical systems for example have such property. However, since this is generally not true memory property have to be assured for objects not having this property.

It is defined in recursive domain with the function,

$$f : N \times X \longrightarrow N \times X$$

$$y^{i+1} = f(x^i) = x^i, \quad n_1 \leq i < n_2.$$

In the language of state transitions it has the ability to assure object position from current state to next state.

This limitation can be removed with memory function control. Let p^i be the value from Bool of decision function at i from N. Then,

$$y^{i+1} = g(p^i, x^i, y^i) = \begin{cases} x^i, \ p^i = 1, & n_1 \leq i < n_2 \\ y^i, \ p^i = 0, & n_1 \leq i < n_2. \end{cases}$$

For $k \geq 1$ and $p^i = 1$ for particular i only, y^{i+k} will have the value of x^i, if $i+k < n_2$.

Extension to represent an object in time domain is straightforward and will not be considered here.

4.3 Function representation in recursive domain

System's behaviour can be represented in recursive domain when its functions are transformed to this domain. Similar extension in representation can be developed for time domain, continous or discrete. This allows synthesis in time domain.

4 Recall a no object situation described in section 2.

Let $f : X_1 x \dots x X_m \longrightarrow Y$ be a function to be represented in recursive domain.

Define object interpretation functions,

$h_1 : N \times X_1 \times \dots \times X_m \longrightarrow X_1 \times \dots \times X_m$

$h_1 (i, x_1, \dots, x_n) = (x'_1, \dots, x'_m)$.

Represent $(i, x_1, \dots, x_m)$ with $(x^i_1, \dots, x^i_m)$, and define

$$h_1 (x^i_1, \dots, x^i_m) = \begin{cases} (x_1, \dots, x_m); & n_1 \leq i < n_2 \\ \text{undef}; & \text{otherwise.} \end{cases}$$

$h_2 : N \times Y \longrightarrow Y$

$h_2 (i, y) = y'$

Represent pair (i,y) with y^i and define,

$$h_2 (y^i) = \begin{cases} y; & n_1 < i \leq n_2 \\ \text{undef}; & \text{otherwise.} \end{cases}$$

Function f can now be represented in recursive domain with function g,

$g : N \times X_1 x \dots x X_m \longrightarrow N \times Y$

$g (i, x_1, \dots, x_m) = (i+1, y)$ or,

$g (x^i_1, \dots, x^i_m) = y^{i+1}$

$$g (x^i_1, \dots, x^i_m) = \begin{cases} f(x_1, \dots, x_m); & n_1 \leq i < n_2 \\ \text{undef}; & \text{otherwise.} \end{cases}$$

Function value is determined with i+1, while domain values are determined with i. This assures memory property. In fact, function g is a composition based on function f and memory function. If defined as controlled function its value can be retained arbitrary long. Figure 4.4 shows controlled object - flow graph for such case.

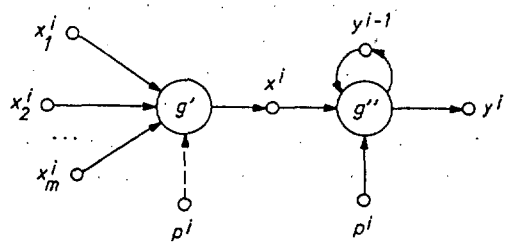


Figure 4.4: Controlled object - flow graph for controlled implementation of g

With the above developments we have minimal tools to define basic structuring concepts.

4.4 Distributive structuring

Assume a system determined with the pairs decision function, controlled function at an arbitrary level of representation. Define a partition of pair set. Each subset of the pair set determines a subsystem, unconnected in general. Define at least one input and at least one output position for each subsystem. Select two arbitrary subsystems and determine all connections between them with regard to unpartitioned system. Between subsystems define transfer functions to reconstruct connections determined with unpartitioned system. At this point representation should be changed to recursive domain in general, since resource sharing is introduced. A sketch of the above development is shown on figure 4.5.

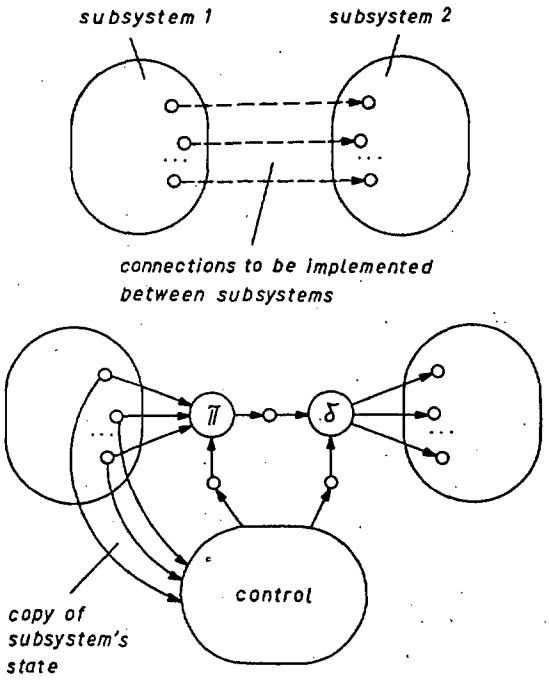
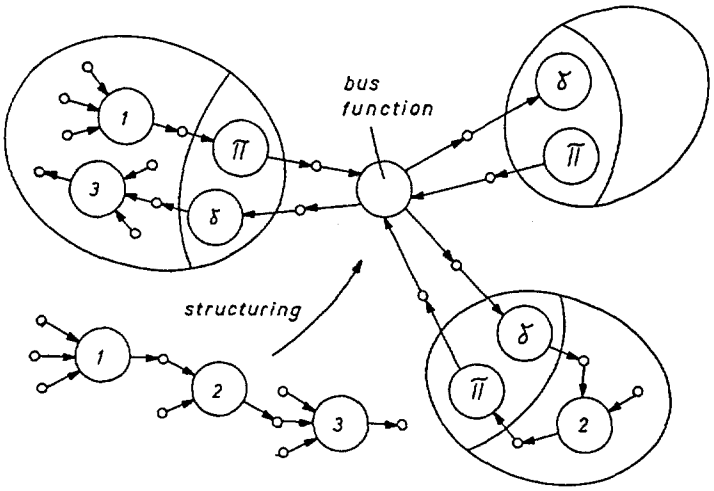


Figure 4.5: A sketch for subsystem connection

The process of developing connections is repeated for all subsystems. Analogous approach is used to develop connections with the system environment. With the insertion of transfer functions which are compositions of selecting, distributive and memory fuctions additional system states are introduced. Decision function which controls particular resource should take this into account to retain compatibility.

Distributive structuring results in a system organized around more or less tightly connected subsystems and preserves the behaviour of the initial system. Figure 4.6 shows a simplified example of a system distributively structured.



Note: Control structure is not shown.

Figure 4.6: Example of distributive structuring

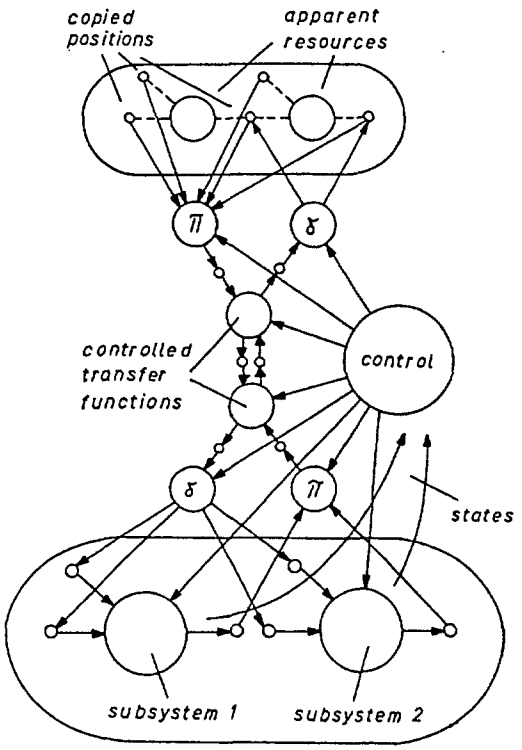


Figure 4.7: Example of hierarchical structuring

4.5 Hierarchical structuring

Assume a system represented with a set of pairs decision function, controlled function. Define a partition, such that each subsystem determined with it is connected. Determine a copy of all those positions, which with the system partition decay to input and output positions. Those positions are actually subsystems' input and output positions. Each copied position will be during structuring process connected to corresponding subsystems' input and output positions.

For each pair consisting of input and output positions determined with the partition and corresponding copied position define transfer function, which enables object transfer from output position over copied position to input position. To connect two subsystems over corresponding copied positions a composition of selecting, memory, and distributive functions is generally needed.

Since objects on copied positions represent system's state at higher level of representation than the initial representation level the behaviour of hierarchically structured system can be interpreted in the context figure 3.5. Transfer functions can namely be completed with object abstraction and interpretation functions. This assures compatibility in representation levels.

The approach to hierarchical structuring can be upgraded to state hierarchy, which is similar to memory hierarchy in contemporary systems.

Similar as with distributive structuring transfer functions introduce additional system's states. Because of this decision functions, which control the execution must be appropriately modified.

Figure 4.7 shows a simplified example of hierarchically structured system.

Both structuring techniques can also be applied on system's control, transfer, and memory structures. Distributed and hierarchical structuring can be combined and applied at arbitrary system representation level. As far as recognized, the proposed structuring techniques are sufficient to model arbitrary system architecture. They were developed for all representations given in section 2. To structure a system in particular representation this can be done without representation change. Since structuring allows resource sharing, system's synthesis can respect cost and performance requirements. Based on the given approach to structuring system's synthesis and software development cannot be separated since they are tightly connected with structuring. This gives at least theoretical possibility for automatic software development. On the other hand highly structured systems can be developed without any control code or software in the usual meaning.

5. CONCLUSION

Review of system's synthesis process is given. Since this topic is very extensive this presentation is focused on these domains estimated as significant. However, domains determined with real - time, fault tolerance, and intelligent behaviour paradigms were completely avoided in the presentation. This does not mean that systems from these domains cannot be developed within the proposed context. In contrary,

some significant practical results were obtained in the synthesis of hard real - time systems and fault tolerance in the domain of industrial process control systems. At the same time it was shown that structuring is still very controverse notion with diverse span of significance, although it is more or less clear that hard real - time and fault tolerance paradigms are of little use in lousy structured domains. Similarly, fault tolerance cannot be a compensation for poor system design. Those were some of the reasons why structuring was given such attention in the presentation.

Since it becomes more clear, that differences caused with separate development of software systems, software engineering, artificial intelligence, knowledge engineering, etc., are caused mainly because of diverse views to problems and that their solution can only be achieved with multidisciplinary approach, latest efforts to avoid such situation result in systems engineering approach.

Based on this approach, systems which are capable to learn particular behaviour, analyse it and construct systems that behave equivalent can be synthesized, based on the proposed approach to the synthesis process.

6. REFERENCES

1 K.M. Kavi, B.P. Buckles, U.N. Bhat
A Formal Definition of Data - Flow Graph Models, IEEE Trans. on Computers, p. 940-948, No. 11, Vol. C-35, Nov. 1986.

2 K.M. Kavi, B.P. Buckles, U.N. Bhat
Isomorphism Between Petri Nets and Dataflow Graphs, IEEE Trans. on Software Eng., p. 1127-1134, No. 10, Vol. SE-13, Oct. 1987.

3 M. Gerkeš
Structures and Models, Resource Interconnection, Functional Behaviour, and Control, Report, Metalna, 1988.

4 M. Gerkeš
Funkcionalno modeliranje sistemov, Strukturiranje, Poročilo, Metalna 1988.

AN INFORMATIONAL THEORY OF DISCOURSE I

INFORMATICA 4/89

Keywords: discourse, discursive environment, discursive process, formalization, information, informational abstraction, informational algebra, informational theory, Lacanian discourse

Anton P. Železnikar*

The research of the discursive nature of information, as determined in [10] and later on in [3, 4, 5, 6], is offered as the property of informing, counter-informing and embedding of information, as its spontaneous arising and cyclicity (circularity). Then, in this respect, informational phenomenology of discourse can be studied as an inherent property of information itself and, afterwards, also as its particularized form, such as is, for instance, the construct of Lacanian discourse. This part of the essay brings a general study of discourse as informational phenomenology and projects this phenomenology onto the Lacanian model of discourse, which is composed of university, master's, hysteric's and analyst's discourse. In the second part of the essay pseudo-Lacanian and other models of discourse will be studied.

Informacijska teorija diskurza I. Raziskava diskurzivne narave informacije, kot je bila opredeljena v [10] in kasneje v [3, 4, 5, 6], se ponuja kot lastnost informiranja, protiinformiranja in vmeščevanja informacije, kot njena spontana nastajalnost in cikličnost (cirkularnost). Informacijsko pojavnost diskurza je tedaj mogoče preučevati z gledišča inherentne lastnosti same informacije, kasneje pa tudi kot njeno partikularizirano obliko, kot je npr. konstrukt lacanovskega diskurza. Ta del spisa prinaša splošno obravnavo diskurza kot informacijske pojavnosti in projicira to pojavnost na model lacanovskega diskurza, ki ga sestavljajo univerzni, gospodarjev, histerikov in analitikov diskurz. V drugem delu spisa bodo obravnavani psevdo-lacanovski in drugi modeli diskurza.

1. INTRODUCTION

> ... disagreement [difference] is the essence of communication. The aberration of sciences ... is that they see the essence of communication in the proper understanding.
>
> Jacques-Alain Miller [1] 41

The term discourse might be understood as personal or interpersonal communication or informing in acts of expressing, talking, uttering, analyzing, conversing, hearing, performing, writing, gesturing, mimicking, signaling, thinking, imagining, etc. In this respect a discourse concerns messaging as well as reception in individual as well as in interindividual arising, exchange, or mediation of information. The discourse can be seen as composed of three parts: the informing of transmitter (informational arising within an informational source), informational mediating (informational propagation or in fact operation between an informational source and informational sink), and informing of receptor (informational sink) considering propagated information. In principle both - the transmitter and the receptor - have the roles of producing and accepting information. But, this is only one side of the meaning we can globally impart to the term discourse. The other side of the meaning has still to be sought in discourse's archaic foundation, i.e. in its Latin origin of the verb dis-curro and the noun discursus.

For our further investigation of possible informational scenarios of discourse the Latin origin of dis-curro and discursus may not be only helpful but also conceptually relevant. The Latin dis-curro has several meanings. It means, for instance, to be full of vivacity (in

* Iskra Delta Computers, Development and Production Center, Stegne 15C, 61000 Ljubljana, Yugoslavia, Europe (or privately: Volaričeva 8, 61000 Ljubljana, Yugoslavia, Europe).

our terms, to be full of informational enterprise or of enterprising informing), to be in high or great spirits, to cause great mirth or to overflow with mirth, or shortly to play, game, perform, animate (inform) spontaneously, according to a being's throwness into a life situation. It means also to disperse, scatter, run different ways, run about, to and fro, to be out of the course, however, returning to it, etc. The Latin discursus has similar meanings.

The modern noun discourse means talk, conversation, discussion, chat, dialogue; speech, address; analysis, dismemberment; soliloquy, monologue, colloque; etc. Further, individual or interpersonal communication may be marked as discursive if it is based on commonsense or logical analysis, discursive thinking or cognition; discursive may mean also notional, logical, deductive, scattered, concentrated, and intuitive. Information, as being defined in [10], is also a discursive phenomenon; simply, information is discursive, it possesses its own informational discursiveness.

The question of informational theory of discourse can concern also epistemological problems, for instance, how a particular discourse represses the truth and perverts the reality. By way of this particular example, we can enter into the domain of the so-called Lacanian types of discourse studying several Lacanian schemata or scenarios of discourse from the informationally theoretic (symbolic, logical [3, 4, 5, 6]) point of view. Certainly, before entering into discursive particularities, we can develop a general theory of discourse which can be particularized into any imaginable form of discursive behavior on the individual and interpersonal level. The basic question is how does a discourse perform informationally or what kind of informing does it perform. Within this context various general and particular scenarios can occur, opening several horizons of possible informational interaction.

2. A GENERAL DEFINITION OF DISCURSIVE, NON-DISCURSIVE, AND ALTERNATIVE ENVIRONMENT

> ... the subject of the unconscious in the Lacan's sense is nothing else than the subject of the marker, this is the scientific subject, which is however marked out in a scientific domain as a discursive subject. This is the subject being always carried by a marker.
>
> Jacques-Alain Miller [1] 64

For the sake of systematics it is possible to distinguish four characteristic cases of discursive and non-discursive environment, which are the following: discursive, non-discursive, alternatively discursive, and alternatively non-discursive environment. For these cases four types (sets) of characteristic informational operators can be introduced, concerning the so-called general, parallel, cyclic, and parallel-cyclic case of discourse, respectively:

S⊨.	$\models, \not\models, \dashv, \not\dashv$
S⊫.	$\Vdash, \not\Vdash, \dashv\!\!\dashv, \not\dashv\!\!\dashv$
S⊢.	$\vdash, \nvdash, \dashv, \not\dashv$
S⊩.	$\Vdash, \not\Vdash, \dashv\!\!\dashv, \not\dashv\!\!\dashv$

It is to point out that all these operators are understood to be particularly discursive or non-discursive, respectively.

2.1. A Discursive Environment

In a general case of discourse we can suppose that several informational sources communicate with several informational sinks or, even more generally, that several informational entities communicate among each other. In a free discourse, where m partners are informationally involved, the basic formula of the discourse can be simply

DE. $\alpha_1, \alpha_2, \ldots, \alpha_m \models \alpha_1, \alpha_2, \ldots, \alpha_m$

This formula represents an informational system consisting of (m by m) parallel informational formulas (informational processes),

DS.
$$\begin{aligned} &\alpha_1 \Vdash \alpha_1;\ \alpha_1 \Vdash \alpha_2;\ \ldots\ \alpha_1 \Vdash \alpha_m;\\ &\alpha_2 \Vdash \alpha_1;\ \alpha_2 \Vdash \alpha_2;\ \ldots\ \alpha_2 \Vdash \alpha_m;\\ &\ldots\ldots\ldots\ldots\ldots\ldots\\ &\alpha_m \Vdash \alpha_1;\ \alpha_m \Vdash \alpha_2;\ \ldots\ \alpha_m \Vdash \alpha_m \end{aligned}$$

The parallel informational operator $\Vdash$ means that the process to which $\Vdash$ belongs can inform each process and can be informed by each process of the system in a parallel way. Simultaneously, $\Vdash$ also means that the process to which it belongs can inform in parallel within itself. If a parallel informational process is formally decomposed, its components can inform explicitly (through explicit parallel informational operators) each other in a parallel way.

The parallel informational system DS shows how a discursive environment DS becomes a net of parallel marked processes. If DS is the metamarking system for DE, then DE is the metamarking discourse (i.e. informational formula) for a "realistically" comprehended act of discourse, etc. In this sense the marking nets on certain levels (or metalevels) are additionally perplexed and conform an integral marking net of understanding of the discourse. In this way, the science of discourse becomes a characteristically marking net within which this science arises and causes changes and arising of the marking net itself.

For a parallel process (PP) $\alpha \Vdash \beta$ the following definition can be introduced:

PP. $(\alpha \Vdash \beta) =_{Df}$
$$(((\exists(\gamma \Vdash \delta)).(\alpha \Vdash \beta;\ \gamma \Vdash \delta)) \vee ((\alpha \Vdash \beta)\ \text{'is\_parallel\_in\_itself'}))$$

This definition can be read in the following way: $\alpha \Vdash \beta$ is a parallel informational process

iff there exists a parallel process $\gamma \Vdash \delta$ such that $\alpha \Vdash \beta$ and $\gamma \Vdash \delta$ belong to a parallel informational system (which means that they interact in parallel with each other) or $\alpha \Vdash \beta$ is parallel (informs parallel) in itself.

In a discursive informational environment DE, informational actors $\alpha_1, \alpha_2, \ldots, \alpha_m$ spontaneously communicate among each other and informationally create cultural (ontological) and individual (metaphysical) forms and processes of information. This is the most general informational model of social discourse as a phenomenon among individual parts (lumps) of a living population. Informational actors impact several actors and are impacted by several ones. The scheme of the parallel discursive system DS shows these possibilities.

The general theory of discourse assumes that within a discursive system the processes of informing are spontaneous, for instance, in the sense of autopoietically structured and organized systems (informational entities). Spontaneity of informing holds on individual as well as populational informational level to the extent to which existing (currently dominating) individual and social informational processes condition and enable various informational modi. In parallel, the similar can be said for the so-called informational cyclicity. In principle, informational processes are circularly structured in their nature of informational arising. The arising itself is a spontaneous and circular process of coming of information into existence. If it is assumed that DS is in principle informationally spontaneous, the question has to be answered how could a DS formally reflect the so-called informational cyclicity. We can set the following definition for a cyclic process:

CP. $(\alpha \vdash \beta) =_{Df}$
$(((\exists(\beta \models \alpha)).(\alpha \models \beta; \beta \models \alpha)) \vee$
$(\alpha, \beta$ 'are_cyclic_in_themselves'$))$

This formula is read as follows: $\alpha \vdash \beta$ is a cyclic process, iff there exists a reflexive process such that processes $\alpha \models \beta$ and $\beta \models \alpha$ belong to an informational system or α and β inform cyclically within themselves.

To explicate both - the parallelism and the cyclicity of a DS - an appropriately structured informational operator $\Vdash$ can be introduced and thus DS can be transformed into the formally adequate form

DS'. $\alpha_1 \Vdash \alpha_1; \alpha_1 \Vdash \alpha_2; \ldots \alpha_1 \Vdash \alpha_m;$
$\alpha_2 \Vdash \alpha_1; \alpha_2 \Vdash \alpha_2; \ldots \alpha_2 \Vdash \alpha_m;$
. . . . . .
$\alpha_m \Vdash \alpha_1; \alpha_m \Vdash \alpha_2; \ldots \alpha_m \Vdash \alpha_m$

For a process $\alpha \Vdash \beta$ there is the following definition:

PC. $(\alpha \Vdash \beta) =_{Df} ((\alpha \models \beta) \wedge (\alpha \vdash \beta))$

This definition says that the process $\alpha \Vdash \beta$ informs parallel and cyclically iff it informs in parallel and simultaneously cyclically. This form of the process offers a rather complex and informationally interwoven situation in which the processes involved can mutually impact and can be impacted in an arbitrarily imaginable and complex manner.

After this discussion it is possible to represent the general formula of discourse in the form

GD. $\alpha \models \beta$

where $\models$ is a particular informational operator of discourse and where α and β represent arbitrary informational sets of informational entities i.e. operands and/or formulas, for instance, $\xi, \eta, \ldots, \zeta$. These entities can be formulas of informational operators and operands, etc. The point of GD is that $\models$ is not a general informational operator but operator of discourse and that α and β are operands being in a discursive relation. Thus, $\alpha \models \beta$ is not a general informational formula but a particular formula concerning the act of discourse.

2.2. A Non-discursive Environment

What happens if informational entities are not in discursive relation? It is certainly possible to express this fact by particular operators giving them the meaning of non-discursive nature. Some problems may occur in defining the so-called non-informational operators, where it is necessary to say explicitly which kind of particularity belongs to a particular operator of non-informing. It is possible to repeat the previous definitions of discursive environment for the case of non-discursiveness of informational processes.

Dually to DE, it is possible to say explicitly that several informational entities do not communicate among each other. The basic formula of a non-discursive environment could be in general

NDE. $\alpha_1, \alpha_2, \ldots, \alpha_m \not\models \alpha_1, \alpha_2, \ldots, \alpha_m$

This formula represents an informational system consisting of (m by m) general informational formulas of non-discursive informing,

NDS. $\alpha_1 \not\models \alpha_1; \alpha_1 \not\models \alpha_2; \ldots \alpha_1 \not\models \alpha_m;$
$\alpha_2 \not\models \alpha_1; \alpha_2 \not\models \alpha_2; \ldots \alpha_2 \not\models \alpha_m;$
. . . . . .
$\alpha_m \not\models \alpha_1; \alpha_m \not\models \alpha_2; \ldots \alpha_m \not\models \alpha_m$

This system may represent a particularly non-discursive environment, where for a case of non-discursive relation $\not\models$ it is possible to determine

ND. $(\alpha \not\models \beta) =_{Df} (\neg(\exists(\alpha, \beta)).(\alpha \models \beta))$

In a similar way it is possible to determine non-discursive relations (informational operators) for cases of parallel, cyclic, and parallel-cyclic processes, respectively:

NPP. $(\alpha \not\Vdash \beta) =_{Df} (\neg(\exists(\alpha, \beta)).(\alpha \Vdash \beta))$

NC. $(\alpha \nvDash \beta) =_{Df} (\neg(\exists(\alpha, \beta)).(\alpha \vdash \beta))$

NCP. $(\alpha \nVdash \beta) =_{Df} (\neg(\exists(\alpha, \beta)).(\alpha \Vdash \beta))$

These cases complete the philosophy concerning the so-called simple or non-alternative cases of discursive and non-discursive processes.

2.3. An Alternatively Discursive Environment

Let the alternatively discursive environment be introduced by saying that in case of a discursive process the act of discourse can happen in one or another way. This means that the possibility of one or another way has to be introduced operationally into formulas describing processes of discourse. One way of discursiveness was presented by the distinguished set of operators $\models$, $\Vdash$, $\vdash$, and $\Vdash$, and their counterparts $\not\models$, $\not\Vdash$, $\not\vdash$, and $\not\Vdash$, denoting the property of non-discursiveness. The other way of discursiveness can be presented by the set of 'opposite' discursive operators $\dashv$, $\dashv\!\!\!\Vdash$, $\dashv\!\!\!\vdash$, and $\dashv\!\!\!\Vdash$, and their counterparts $\not\dashv$, $\not\dashv\!\!\!\Vdash$, $\not\dashv\!\!\!\vdash$, and $\not\dashv\!\!\!\Vdash$, denoting another way of the property of non-discursiveness.

Instead of a simple discursive environment DE it is possible to explicate the alternative environment by the system

ADE. $\alpha_1, \alpha_2, \ldots, \alpha_m \models \alpha_1, \alpha_2, \ldots, \alpha_m;$
$\alpha_1, \alpha_2, \ldots, \alpha_m \dashv \alpha_1, \alpha_2, \ldots, \alpha_m$

This system says that informational sources and sinks $\alpha_1, \alpha_2, \ldots, \alpha_m$ communicate among each other in one ($\models$) or another way ($\dashv$). If this communication occurs in a parallel way, the parallel decomposed system is

ADS.
$$\begin{array}{l}
\alpha_1 \Vdash \alpha_1;\ \alpha_1 \Vdash \alpha_2;\ \ldots\ \alpha_1 \Vdash \alpha_m;\\
\alpha_2 \Vdash \alpha_1;\ \alpha_2 \Vdash \alpha_2;\ \ldots\ \alpha_2 \Vdash \alpha_m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\\
\alpha_m \Vdash \alpha_1;\ \alpha_m \Vdash \alpha_2;\ \ldots\ \alpha_m \Vdash \alpha_m;\\
\alpha_1 \dashv\!\!\!\Vdash \alpha_1;\ \alpha_1 \dashv\!\!\!\Vdash \alpha_2;\ \ldots\ \alpha_1 \dashv\!\!\!\Vdash \alpha_m;\\
\alpha_2 \dashv\!\!\!\Vdash \alpha_1;\ \alpha_2 \dashv\!\!\!\Vdash \alpha_2;\ \ldots\ \alpha_2 \dashv\!\!\!\Vdash \alpha_m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\\
\alpha_m \dashv\!\!\!\Vdash \alpha_1;\ \alpha_m \dashv\!\!\!\Vdash \alpha_2;\ \ldots\ \alpha_m \dashv\!\!\!\Vdash \alpha_m
\end{array}$$

Similarly to PP, CP, PC, and GD it is possible to define the following alternative cases, respectively:

APP. $(\beta \dashv\!\!\!\Vdash \alpha) =_{Df}$
$(((\exists(\delta \dashv\!\!\!\Vdash \gamma)).(\beta \dashv\!\!\!\Vdash \alpha;\ \delta \dashv\!\!\!\Vdash \gamma)) \vee$
$((\beta \dashv\!\!\!\Vdash \alpha)\ \text{'is\_parallel\_in\_itself'}))$

ACP. $(\beta \dashv\!\!\!\vdash \alpha) =_{Df}$
$(((\exists(\alpha \dashv \beta)).(\beta \dashv \alpha;\ \alpha \dashv \beta)) \vee$
$(\alpha, \beta\ \text{'are\_cyclic\_in\_themselves'}))$

APC. $(\beta \dashv\!\!\!\vdash \alpha) =_{Df} ((\beta \dashv\!\!\!\Vdash \alpha) \wedge (\beta \dashv\!\!\!\vdash \alpha))$

AGD. $\beta \dashv \alpha$

If entities α and β are in a process of alternative discourse, the formula

AD. $(\alpha \models \beta) \vee (\beta \dashv \alpha)$

means that α informs discursively β in one or another way or that β is informed discursively in one or another way.

2.4. An Alternatively Non-discursive Environment

Which kind of environment is alternatively non-discursive? Does a kind of totally non-discursive living environment exist at all? It is possible to construct such an environment abstractly, however only particularly, that is by introducing particular types of non-discursive operators. One kind or particularism of non-discursiveness does not mean that there does not exist or arise another type of discursiveness of the observed informational (discursive) entity. We have already pointed out some typical dilemmas of non-informing (non-discursiveness).

Dually to ADE it is possible to explicate the so-called alternatively non-discursive environment by the system

ANE. $\alpha_1, \alpha_2, \ldots, \alpha_m \not\models \alpha_1, \alpha_2, \ldots, \alpha_m;$
$\alpha_1, \alpha_2, \ldots, \alpha_m \not\dashv \alpha_1, \alpha_2, \ldots, \alpha_m$

This system says that informational sources and sinks $\alpha_1, \alpha_2, \ldots, \alpha_m$ do not communicate among each other in any way (neither $\models$ nor $\dashv$). This kind of non-informing can be expressed by the marking net of the form

ANS.
$$\begin{array}{l}
\alpha_1 \not\Vdash \alpha_1;\ \alpha_1 \not\Vdash \alpha_2;\ \ldots\ \alpha_1 \not\Vdash \alpha_m;\\
\alpha_2 \not\Vdash \alpha_1;\ \alpha_2 \not\Vdash \alpha_2;\ \ldots\ \alpha_2 \not\Vdash \alpha_m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\\
\alpha_m \not\Vdash \alpha_1;\ \alpha_m \not\Vdash \alpha_2;\ \ldots\ \alpha_m \not\Vdash \alpha_m;\\
\alpha_1 \not\dashv\!\!\!\Vdash \alpha_1;\ \alpha_1 \not\dashv\!\!\!\Vdash \alpha_2;\ \ldots\ \alpha_1 \not\dashv\!\!\!\Vdash \alpha_m;\\
\alpha_2 \not\dashv\!\!\!\Vdash \alpha_1;\ \alpha_2 \not\dashv\!\!\!\Vdash \alpha_2;\ \ldots\ \alpha_2 \not\dashv\!\!\!\Vdash \alpha_m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\\
\alpha_m \not\dashv\!\!\!\Vdash \alpha_1;\ \alpha_m \not\dashv\!\!\!\Vdash \alpha_2;\ \ldots\ \alpha_m \not\dashv\!\!\!\Vdash \alpha_m
\end{array}$$

It is important to stress that operators (in fact metaoperators) $\not\Vdash$ and $\not\dashv\!\!\!\Vdash$ in distinct processes of the system ANS can be marked by mutually different operational particularizations, i.e. by operational markers which mark different acts of non-discursive informing. In this manner each alternatively non-discursive environment is non-discursive only to the extent of certain particularities, and thus can never be absolutely or totally non-discursive. This can be immediately understood on the formal or on the marking level, if metasystem ANS is particularized in the following way:

$$\begin{array}{l}
\alpha_1 \not\Vdash_{\alpha} \alpha_1;\ \alpha_1 \not\Vdash_{\beta} \alpha_2;\ \ldots\ \alpha_1 \not\Vdash_{\gamma} \alpha_m;\\
\alpha_2 \not\Vdash_{\lambda} \alpha_1;\ \alpha_2 \not\Vdash_{\mu} \alpha_2;\ \ldots\ \alpha_2 \not\Vdash_{\nu} \alpha_m;\\
\ldots\ldots\ldots\ldots\ldots\ldots
\end{array}$$

$$\alpha_m \not\models_\xi \alpha_1;\ \alpha_m \not\models_\eta \alpha_2;\ \dots\ \alpha_m \not\models_\zeta \alpha_m;$$
$$\alpha_1 \not\models_\alpha \alpha_1;\ \alpha_1 \not\models_\beta \alpha_2;\ \dots\ \alpha_1 \not\models_\gamma \alpha_m;$$
$$\alpha_2 \not\models_\lambda \alpha_1;\ \alpha_2 \not\models_\mu \alpha_2;\ \dots\ \alpha_2 \not\models_\nu \alpha_m;$$
$$\dots\dots\dots\dots\dots\dots$$
$$\alpha_m \not\models_\xi \alpha_1;\ \alpha_m \not\models_\eta \alpha_2;\ \dots\ \alpha_m \not\models_\zeta \alpha_m$$

A particular non-discursiveness of a process has the meaning of non-communication or of the lack of a particular understanding among distinct informational entities. It seems to be important to have explicit possibilities for the expression of different forms concerning non-understanding in discursive processes.

Similarly to ND, NPP, NC, and NCP, it is possible to define the following alternative cases, respectively:

AND. $(\beta \not\models \alpha) =_{Df} (\neg(\exists(\beta, \alpha)).(\beta \dashv \alpha))$

ANP. $(\beta \not\models\!\!\models \alpha) =_{Df} (\neg(\exists(\beta, \alpha)).(\beta \dashv\!\!\dashv \alpha))$

ANC. $(\beta \not\dashv \alpha) =_{Df} (\neg(\exists(\beta, \alpha)).(\beta \dashv \alpha))$

ANCP. $(\beta \not\dashv\!\!\dashv \alpha) =_{Df} (\neg(\exists(\beta, \alpha)).(\beta \dashv\!\!\dashv \alpha))$

This completes the discussion concerning the alternatively non-discursive environment.

3. DISCOURSE AS INFORMING IN ITSELF

> ... It is possible to postulate anything, however, the value of the obtained mathematics will be showed by its applications.
>
> Zvonimir Šikić [2] 32

3.1. General Informing within a Discourse of an Informational Entity

What is in fact a discourse in itself? Is it a sort of communication in which an informational entity communicates with itself or, more precisely, informs itself? Already in the discursive system DS formulas of the form $\alpha \models \alpha$ appeared; do they represent the process of discourse in itself? If so, then it would be possible to decompose the process $\alpha \models \alpha$ in at least two components which would mark the "speaking" component against the "addressed" one in this process.

It is certainly possible to suppose that an informational entity is always in the relation to be discursive in itself or to itself. This fact could simply denote the nature of information and its informing as living, artificial, or cosmic phenomenology. The discursive nature of informational entity α could be logically postulated by the formula or system of two simple formulas, i.e.,

DNα. $(\alpha \models) \vee (\models \alpha)$ or
$\alpha \models;\ \models \alpha$

respectively. This formula or system characterizes the entity α as to be discursive. The meaning of this formula is that α informs and/or is informed discursively. Within this formula, $\models$ appears as a unary informational operator of discourse, which in a concrete situation can be adequately particularized, to mark the desired case of discursive informing of the entity α. It is possible to say that formulas $\alpha \models$ and $\models \alpha$ are discursively open formulas. In general, formulas using unary operators are always particularly open or generally unclosed.

Formula and system in DNα can be even more general, if it is said that α informs and/or is informed discursively in one and/or another way. In this case the basic discursive nature of entity α can be expressed as

GNα. $(\alpha \models) \vee (\models \alpha) \vee (\dashv \alpha) \vee (\alpha \dashv)$ or
$\alpha \models;\ \models \alpha;\ \dashv \alpha;\ \alpha \dashv$

This case argues the introduction of the symmetric discursive operator $\dashv$.

The conditional (implication) of formula DNα is certainly

CNα. $((\alpha \models) \vee (\models \alpha)) \Rightarrow (\alpha \models \alpha)$

If, in general, information α discursively informs or is informed, then information α is in a discursive relation to itself. However, this does not mean that information α is not simultaneously in a discursive relation with other information β, i.e.,

CNβ. $((\alpha \models) \vee (\models \alpha)) \Rightarrow_\pi ((\alpha \models \beta) \vee (\beta \models \alpha))$

If, in general, information α discursively informs or is informed, then it is possible that information α informs other information β or is informed by other information β. The operator $\Rightarrow_\pi$ denotes the so-called possible conditional case.

3.2. Counter-informing within a Discourse of an Informational Entity

The next question of the self-discursive process $\alpha \models \alpha$ can be the following: how does information α discursively inform itself? In a discourse, information α arises as counter-information ω, which has to be embedded into the source or originally existing information α. We can simply say that in a self-discursive game (informing), the discourse within α, marked by δ_α, arises or develops as its own counter-discourse δ_ω, embracing counter-informational components $\complement$ and ω. This counter-discourse has to be discursively embedded by δ_ε into the original discourse δ_α.

Up to now the only scended formula of a self-discourse was $\alpha \models \alpha$. It is possible to connect the so-called classical informational components, i.e., information α, its informing $\Im$, counter-informing $\complement$, counter-information ω, informational embedding $\mathfrak{E}$, and embedding information ε, with discourse δ_α, counter-discourse δ_ω, and discursive embedding δ_ε. It is possible to postulate the following system of equivalences:

Dα. $\delta_\alpha \equiv (\alpha \models \alpha)$;
$\delta_\omega \equiv (\delta_\alpha \models \mathfrak{C}) \models \omega$;
$\delta_\varepsilon \equiv (\delta_\omega \models \mathfrak{E}) \models \varepsilon$

Between the discursive and informational components the following correspondences can be observed:

DαR. $\delta_\alpha \leftrightarrow \alpha, \Im$; $\delta_\omega \leftrightarrow \mathfrak{C}, \omega$: $\delta_\varepsilon \leftrightarrow \mathfrak{E}, \varepsilon$

How is it possible to postulate the process of the counter-informing $\mathfrak{C}$ by which the counter-information ω is coming into existence? How does this process begin to arise? Let us introduce two particular informational operators for marking the looming (bursting) of this process in one or another way:

L$\mathfrak{C}\omega$. $(\alpha \models \alpha) \llcorner$
$(\alpha \llcorner \mathfrak{C}; \mathfrak{C} \lrcorner \alpha; \mathfrak{C} \llcorner \omega; \omega \lrcorner \mathfrak{C})$

This system of four processes in the second line has to be understood as the beginning of the arising (operators $\llcorner$ and $\lrcorner$) of counter-discourse δ_ω (i.e. $\mathfrak{C}$, ω) out of discourse δ_α (i.e. $\alpha \models \alpha$). Certainly, the process of counter-informing $\mathfrak{C}$ has its beginning (looming). The last formula can be read as follows: the discourse $\alpha \models \alpha$ looms the counter-discursive processing $\mathfrak{C}$ and in parallel (simultaneously) $\mathfrak{C}$ looms the counter-information ω in one or another way. In fact, these four processes constitute the parallel counter-discursive system δ_ω. To stress the parallelness of these processes after the looming of $\mathfrak{C}$ out of α and after the looming of ω out of $\mathfrak{C}$, in the next step the following (discursively regular) formula can be introduced:

P$\mathfrak{C}\omega$. $\delta_\omega \equiv$
$((\alpha \models \alpha) \Vdash$
$(\alpha \Vdash \mathfrak{C}; \mathfrak{C} \dashv\!\Vdash \alpha; \mathfrak{C} \Vdash \omega; \omega \dashv\!\Vdash \mathfrak{C}))$

It is possible to interpret the operators $\llcorner$ and $\lrcorner$ in the primordial process L$\mathfrak{C}\omega$ as particularizations of the parallel metaoperators $\Vdash$ and $\dashv\!\Vdash$, respectively. The last formula can be read in the following way: the discourse $\alpha \models \alpha$ informs the counter-informing $\mathfrak{C}$ in parallel in one or another way and the counter-informing $\mathfrak{C}$ informs in parallel the counter-information ω in one or another way. It can be seen that in these processes there are not processes which could constitute the condition of the so-called discursive cycle. So, the process of counter-discourse δ_ω is discursively open. It only means that further discursive (informational) processes have to be added to the given system to establish the circumstances of discursive circularity. It is even reasonable to join the counter-looming and counter-informing system in a unique counter-discursive system, for looming of counter-discourse is a steady process within a flowing process of discourse. More formulas in such a system only means that more particular information concerning discourse is on disposal. The complex game of counter-discursiveness and informational embedding concerning the discourse as a whole will be formulated in the informationally cyclic form in section 3.4.

3.3. Informational Embedding within a Discourse of an Informational Entity

The next question which arises is what to do with the so-called counter-discourse or how to bring it into the context of a developing discourse. The "interest" or intention of a discourse could be to capture meaningfully as much as possible of the arisen counter-discourse, with the goal to get some origins for further development of discourse. It seems reasonable to separate or decompose this particular process of embedding, which arises in the dynamic environment of the developing discourse.

Discursive embedding, marked by δ_ε is a part of the so-called discursive cycle. This cycle can be formally expressed in the following way:

DCω. $((\delta_\alpha \models \delta_\omega) \models \delta_\varepsilon) \models \delta_\alpha$

This formula is important for the understanding of the δ_ε's role when δ_ε produces the so-called embedding information ε, by which counter-information ω is informationally embedded or connected to the source information α. Certainly, embedding information ε does not necessarily offer the complete embedding or connectedness of ω in regard to α, but ensures that counter-informational result ω is not lost in the process of discursive informing.

Discursive embedding δ_ε as informational phenomenon underlies the process of looming of the embedding discourse and its regular continuation, for instance, in the form of an adequate parallel informational system. First, the following process of the looming of discursive embedding can be assumed:

L$\mathfrak{E}\varepsilon$. $((\alpha \models \alpha) \llcorner$
$(\alpha \llcorner \mathfrak{C}; \mathfrak{C} \lrcorner \alpha; \mathfrak{C} \llcorner \omega; \omega \lrcorner \mathfrak{C})) \llcorner$
$(\omega \llcorner \mathfrak{E}; \mathfrak{E} \lrcorner \omega; \mathfrak{E} \llcorner \varepsilon; \varepsilon \lrcorner \mathfrak{E})$

This system with four processes in the third line has to be understood as the beginning of the arising of the embedding discourse δ_ε (i.e., $\mathfrak{E}$, ε) out of discourse δ_ω (i.e., $\mathfrak{C}$, ω), when δ_ω begins out of δ_α (i.e., $\alpha \models \alpha$). The last formula can be read as follows: the discursive process $\alpha \models \alpha$ looms the looming of the counter-discursive process, where counter-informing $\mathfrak{C}$ is loomed in one or another way by information α and counter-information ω is loomed in one or another way by counter-informing $\mathfrak{C}$, and then these two discursive processes loom the looming of the embedding discursive process, where counter-information ω loomed in the counter-discursive process looms embedding $\mathfrak{E}$ in one or another way and embedding $\mathfrak{E}$ looms embedding information ε in one or another way.

In fact, four processes in the third line of L$\mathfrak{E}\varepsilon$ constitute the beginning of the process of

discursive embedding $\delta_{\mathfrak{E}}$. This process can be understood to be completely parallel, thus, it can be adequately expressed in the form

$$P\mathfrak{E}\varepsilon. \quad \delta_{\mathfrak{E}} \equiv (((\alpha \models \alpha) \models (\alpha \models \mathfrak{C}; \mathfrak{C} \dashv\models \alpha; \mathfrak{C} \models \omega; \omega \dashv\models \mathfrak{C})) \models (\omega \models \mathfrak{E}; \mathfrak{E} \dashv\models \omega; \mathfrak{E} \models \varepsilon; \varepsilon \dashv\models \mathfrak{E}))$$

The last formula, which was logically deduced from $L\mathfrak{E}\varepsilon$ by universalizing operators $\llcorner$ and $\lrcorner$ by operators $\models$ and $\dashv\models$, can be read in the following way: embedding discourse $\delta_{\mathfrak{E}}$ is constituted by counter-discourse δ_{ω}, which parallel informs the four characteristic parallel processes of embedding of counter-information ω, concerning informational embedding $\mathfrak{E}$ and embedding information ε (as shown by the third line of $P\mathfrak{E}\varepsilon$). In short, $P\mathfrak{E}\varepsilon$ can be rewritten into

$$\delta_{\mathfrak{E}} \equiv (\delta_{\omega} \models (\omega \models \mathfrak{E}; \mathfrak{E} \dashv\models \omega; \mathfrak{E} \models \varepsilon; \varepsilon \dashv\models \mathfrak{E}))$$

where

$$\delta_{\omega} \equiv (\delta_{\alpha} \models (\alpha \models \mathfrak{C}; \mathfrak{C} \dashv\models \alpha; \mathfrak{C} \models \omega; \omega \dashv\models \mathfrak{C})); \quad \delta_{\alpha} \equiv (\alpha \models \alpha)$$

It could be said that the last three expressions are in accordance with the equivalence system $D\alpha$.

However, formula $P\mathfrak{E}\varepsilon$ does not say how or where the counter-discourse δ_{ω} will be embedded by means of embedding discourse $\delta_{\mathfrak{E}}$. This answer will be given in the next section.

3.4. The Game of Informing, Counter-informing, and Informational Embedding within a Discourse of an Informational Entity

The course of discourse within an informational entity depends essentially from the game in which informing, counter-informing, and informational embedding take part as substantial informational players. This game is circular in the sense that after the looming of discourse δ_{α}, this is closed via counter-discourse δ_{ω} and embedding discourse $\delta_{\mathfrak{E}}$ into the so-called discursive cycle. This cycle was already described by formula $DC\omega$ in the previous section.

The game of discursive looming as the beginning of the game of discourse can be described according to $L\mathfrak{C}\omega$ and $L\mathfrak{E}\varepsilon$ and considering $D\alpha\omega$ by

$$LD\alpha. \quad (((\alpha \models \alpha) \llcorner (\alpha \llcorner \mathfrak{C}; \mathfrak{C} \lrcorner \alpha; \mathfrak{C} \llcorner \omega; \omega \lrcorner \mathfrak{C})) \llcorner (\omega \llcorner \mathfrak{E}; \mathfrak{E} \lrcorner \omega; \mathfrak{E} \llcorner \varepsilon; \varepsilon \lrcorner \mathfrak{E})) \llcorner (\varepsilon \llcorner \alpha; \alpha \lrcorner \varepsilon; \alpha \models \alpha)$$

This formula is cyclic within the basic process $\alpha \models \alpha$. The last line of the formula can be read in the following way: embedding information ε, which carries information on arisen counter-information ω, looms into source information α in one or another way and, thus, informationally impacts the basic process of discourse $\alpha \models \alpha$. To remain consequent in the relation of possibility of decomposition, the basic process $\alpha \models \alpha$ could be replaced by the cyclic system

$$\alpha \models \mathfrak{I}; \mathfrak{I} \dashv\models \alpha; \mathfrak{I} \models \alpha; \alpha \dashv\models \mathfrak{I}$$

or in the case of looming by

$$\alpha \llcorner \mathfrak{I}; \mathfrak{I} \lrcorner \alpha; \mathfrak{I} \llcorner \alpha; \alpha \lrcorner \mathfrak{I}$$

Probably, the last interpretation can satisfy the taste of a theorist's view for it does not limit in any respect the possibility of further development of formal treating of informational phenomenology in question.

The next step in the cyclic game of discourse is the well-known transition from the process of looming into the process of parallel informing. Thus, $LD\alpha$ becomes

$$PD\alpha. \quad (((\alpha \models \alpha) \models (\alpha \models \mathfrak{C}; \mathfrak{C} \dashv\models \alpha; \mathfrak{C} \models \omega; \omega \dashv\models \mathfrak{C})) \llcorner (\omega \models \mathfrak{E}; \mathfrak{E} \dashv\models \omega; \mathfrak{E} \models \varepsilon; \varepsilon \dashv\models \mathfrak{E})) \llcorner (\varepsilon \models \alpha; \alpha \dashv\models \varepsilon; \alpha \models \alpha)$$

This formula images the self-discursive game within informational entity α. In this formula, entity ε functions as the resulting backward information concerning the discourse within an informational entity α. Through closing of the discursive cycle, partial discursive components δ_{α}, δ_{ω}, and $\delta_{\mathfrak{E}}$, described previously as non-cyclic components, can get a new, dynamic meaning. And this is the case explicated in $PD\alpha$ in respect to $DC\omega$. It is believed that according to the previous discussion the reader could be capable to develop autonomously any connective information (or formal proving) if necessary.

4. DISCOURSE AS INTERINFORMATIONAL INFORMING

> ... Pragmatic mathematics (which in fact is everyday, standard mathematics) plunges through its applying into experimental sciences and, in the last consequence, through them can be experimentally proven or disproven.
>
> Zvonimir Šikić [2] 32

4.1. General Informing within a Discourse among Several Informational Entities

The discourse between two entities (for instance, existent things, individuals, informational items, etc.) has to be understood always as composed of two types of processes: the inter-entities' and the self-entity's one. It means that each entity discursively involved performs the interinformational and self-informational informing simultaneously. Thus, formulas of self-informational informing

within a discourse remain valid also within an interinformational discursive process.

What in fact is a discourse between two informational entities? It is a kind of communication in which entities communicate with themselves and each of them with the other one. If so, it is necessary to study the basic discursive process

B1. $\alpha, \beta \models \alpha, \beta$

in detail, considering that α as well as β is "speaking" as well as "addressed" component simultaneously. To study processes in detail, within informational logic, means to develop more and more detailed formulas and join them to the initial informational system.

A further generalization of the discourse can be studied starting by the discursive formula

B2. $\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$

where particular discursive processes among entities $\alpha, \beta, \ldots, \gamma$ take place. Formula B2 enables the two-way discourse among all informational entities $(\alpha, \beta, \ldots, \gamma)$, occurring on the left and on the right side of the formula.

It is worth mentioning that the one-way discourse between entities α and β is possible, denoting this initially by

B3. $\alpha \models \beta$

In this case α remains always the transmitter and β always the receptor of α's messages. In this relation, α and β remain discursive within themselves, but only α transmits information to β while β remains against α a pure informational receptor. In the two-way process $\alpha, \beta \models \alpha, \beta$, transmitting and receiving roles of α and β are interchanging, so, both of them can function as the transmitter and receptor. A more general one-way discourse can be expressed by the formula

B4. $\alpha, \beta, \ldots, \gamma \models \xi, \eta, \ldots, \zeta$

where entities $\alpha, \beta, \ldots, \gamma, \xi, \eta, \ldots, \zeta$ mark the pairwise different entities and where entities $\alpha, \beta, \ldots, \gamma$ function as transmitters and $\xi, \eta, \ldots, \zeta$ as receptors.

The so-called self-discursiveness of an informational entity α was logically postulated by DNα through the scheme $(\alpha \models) \vee (\models \alpha)$. But, this formula does not concern merely the self-discursiveness, for it is an open formula (by the use of unary operators $\models$, which are always open to the other side) and thus can communicate not only to itself, but also to any other informational entity. In fact, $\alpha \models$ is to be understood as the formula

B5. $\alpha \models \alpha, \beta, \ldots, \gamma$

which on the right side of $\models$ is not limited by distinct informational entities, postulating

B6. $(\alpha \models) \Rightarrow_{\pi} (\alpha \models \alpha, \beta, \ldots, \gamma)$

Similarly, $\models \alpha$ is to be understood as

B7. $\alpha, \beta, \ldots, \gamma \models \alpha$

postulating

B8. $(\models \alpha) \Rightarrow_{\pi} (\alpha, \beta, \ldots, \gamma \models \alpha)$

Formulas B6 and B8 can be expressed in a general form, if it is said that α informs and/or is informed discursively in one or another way. In this case, B6 and B8 become

B9. $((\alpha \models) \vee (\dashv \alpha)) \Rightarrow_{\pi}$
$((\alpha \models \alpha, \beta, \ldots, \gamma) \vee$
$(\alpha, \beta, \ldots, \gamma \dashv \alpha))$

B10. $((\models \alpha) \vee (\alpha \dashv)) \Rightarrow_{\pi}$
$((\alpha, \beta, \ldots, \gamma \models \alpha) \vee$
$(\alpha \dashv \alpha, \beta, \ldots, \gamma))$

respectively.

4.2. Counter-informing within a Discourse among Several Informational Entities

In this section the following basic forms of discursive informing will be examined:

B11. $\alpha \models \beta$,
$\alpha, \beta, \ldots, \gamma \models \xi, \eta, \ldots, \zeta$,
$\alpha, \beta \models \alpha, \beta$, and
$\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$

The first two cases denote the so-called one-way informing and the last two cases the two-way one. According to these cases, the following notations of appearing discursive components can be introduced:

B12. $\delta(\alpha)$, $\delta(\alpha, \omega)$, and $\delta(\alpha, \varepsilon)$

These entities mark α's discursive components as described by Dα, within which $\mathbb{C}(\alpha)$, $\omega(\alpha)$, $\mathbb{E}(\alpha)$, and $\varepsilon(\alpha)$ appear as counter-informing, counter-information, informational embedding, and embedding information, respectively. Further, α marks any operand-informational entity in the upper cases, so,

$$\alpha \in \{\alpha, \beta, \ldots, \gamma, \xi, \eta, \ldots, \zeta\}$$

According to Dα, the following self-discursive equivalences, called α's self-discourse, counter-discourse, and embedding discourse, marked by

B13. $\delta(\alpha) \equiv (\alpha \models \alpha)$;
$\delta(\alpha, \omega) \equiv (\delta(\alpha) \models \mathbb{C}(\alpha)) \models \omega(\alpha)$;
$\delta(\alpha, \varepsilon) \equiv (\delta(\alpha, \omega) \models \mathbb{E}(\alpha)) \models \varepsilon(\alpha)$

can be introduced, respectively, for $\alpha \in \{\alpha, \beta, \ldots, \gamma, \xi, \eta, \ldots, \zeta\}$. Together with these equivalences, the following inter-discursive cases, called $(\alpha \models \beta)$'s discourse, counter-discourse, and embedding discourse, marked by

B14.
$\delta(\alpha \models \beta) \equiv (\alpha \models \beta)$;
$\delta(\alpha \models \beta; \omega) \equiv (\delta(\alpha \models \beta) \models \mathbb{C}(\alpha \models \beta)) \models \omega(\alpha \models \beta)$;
$\delta(\alpha \models \beta; \varepsilon) \equiv (\delta(\alpha \models \beta; \omega) \models \mathbb{E}(\alpha \models \beta)) \models$
$\varepsilon(\alpha \models \beta)$

can be introduced, respectively, for pairwise different α and β, where $\alpha, \beta \in \{\alpha, \beta, \dots, \gamma, \xi, \eta, \dots, \zeta\}$. Of course, it can happen that some of these particular discourses do not appear or, as it is said, are void. As we see, B14 is only a particular case of B13, if α in B13 marks any informational entity.

4.2.1. The Counter-discursive Case $\alpha \models \beta$

Informational process, marked by $\alpha \models \beta$, has its own, characteristically (one-way) shaped counter-discursiveness. The initial question is: how does the phenomenon of counter-discourse within $\alpha \models \beta$ begin? At the beginning, there is the looming (or bursting) of all possible forms of counter-informing processes $\mathfrak{C}(\alpha)$, $\mathfrak{C}(\beta)$, and $\mathfrak{C}(\alpha \models \beta)$ and corresponding counter-informational products $\omega(\alpha)$, $\omega(\beta)$, and $\omega(\alpha \models \beta)$, produced by counter-informing processes in one or another way. Similarly to L$\mathfrak{C}\omega$ there is

B15.

$$(\alpha \models \beta) \llcorner$$
$$(\alpha \llcorner \mathfrak{C}(\alpha);\ \mathfrak{C}(\alpha) \lrcorner \alpha;\ \mathfrak{C}(\alpha) \llcorner \omega(\alpha);\ \omega(\alpha) \lrcorner \mathfrak{C}(\alpha);$$
$$\beta \llcorner \mathfrak{C}(\beta);\ \mathfrak{C}(\beta) \lrcorner \beta;\ \mathfrak{C}(\beta) \llcorner \omega(\beta);\ \omega(\beta) \lrcorner \mathfrak{C}(\beta);$$
$$(\alpha \models \beta) \llcorner \mathfrak{C}(\alpha \models \beta);\ \mathfrak{C}(\alpha \models \beta) \lrcorner (\alpha \models \beta);$$
$$\mathfrak{C}(\alpha \models \beta) \llcorner \omega(\alpha \models \beta);\ \omega(\alpha \models \beta) \lrcorner \mathfrak{C}(\alpha \models \beta))$$

This formula includes four counter-informational processes for each entity α, β, and $\alpha \models \beta$, respectively, and describes the beginning of the arising of counter-discourses $\delta(\alpha, \omega)$, $\delta(\beta, \omega)$, and $\delta(\alpha \models \beta, \omega)$ out of discourse $\delta(\alpha \models \beta)$. After the occurrence of looming, the looming processes of counter-informing pass over to their resulting regular parallel forms, thus, to the resulting counter-discourse $\delta_r(\alpha \models \beta, \omega)$ within $\alpha \models \beta$:

B16.

$$\delta_r(\alpha \models \beta;\ \omega) \equiv$$
$$((\alpha \models \beta) \Vdash$$
$$(\alpha \Vdash \mathfrak{C}(\alpha);\ \mathfrak{C}(\alpha) \nVdash \alpha;\ \mathfrak{C}(\alpha) \Vdash \omega(\alpha);\ \omega(\alpha) \nVdash \mathfrak{C}(\alpha);$$
$$\beta \Vdash \mathfrak{C}(\beta);\ \mathfrak{C}(\beta) \nVdash \beta;\ \mathfrak{C}(\beta) \Vdash \omega(\beta);\ \omega(\beta) \nVdash \mathfrak{C}(\beta);$$
$$(\alpha \models \beta) \Vdash \mathfrak{C}(\alpha \models \beta);\ \mathfrak{C}(\alpha \models \beta) \nVdash (\alpha \models \beta);$$
$$\mathfrak{C}(\alpha \models \beta) \Vdash \omega(\alpha \models \beta);\ \omega(\alpha \models \beta) \nVdash \mathfrak{C}(\alpha \models \beta)))$$

So far, the last formula completes the discussion concerning the counter-discursive component of the process $\alpha \models \beta$.

4.2.2. The Counter-discursive Case $\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta$

This case represents the most general, inductively broadened one-way discourse among informational transmitters $\alpha, \beta, \dots, \gamma$ and receptors $\xi, \eta, \dots, \zeta$. Thus, the discussion from section 4.2.1 can be repeated in a general way.

At the beginning, there is the looming (or bursting) of all possible forms of counter-informing processes

B17.

$$\mathfrak{C}(\varphi),\ \varphi \in \{\alpha, \beta, \dots, \gamma, \xi, \eta, \dots, \zeta\};$$
$$\mathfrak{C}(\psi \models \tau),\ \psi \in \{\alpha, \beta, \dots, \gamma\},\ \tau \in \{\xi, \eta, \dots, \zeta\}$$

and the corresponding counter-informational products

B18.

$$\omega(\varphi),\ \varphi \in \{\alpha, \beta, \dots, \gamma, \xi, \eta, \dots, \zeta\};$$
$$\omega(\psi \models \tau),\ \psi \in \{\alpha, \beta, \dots, \gamma\},\ \tau \in \{\xi, \eta, \dots, \zeta\}$$

produced by counter-informing processes in one or another way. Similarly to B15 there is

B19.

$$(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta) \llcorner$$
$$((\exists(\varphi \in \{\alpha, \beta, \dots, \gamma, \xi, \eta, \dots, \zeta\}).$$
$$(\varphi \llcorner \mathfrak{C}(\varphi);\ \mathfrak{C}(\varphi) \lrcorner \varphi;\ \mathfrak{C}(\varphi) \llcorner \omega(\varphi);$$
$$\omega(\varphi) \lrcorner \mathfrak{C}(\varphi)));$$
$$(\exists(\psi \in \{\alpha, \beta, \dots, \gamma\}, \tau \in \{\xi, \eta, \dots, \zeta\}).$$
$$((\psi \models \tau) \llcorner \mathfrak{C}(\psi \models \tau);\ \mathfrak{C}(\psi \models \tau) \lrcorner (\psi \models \tau);$$
$$\mathfrak{C}(\psi \models \tau) \llcorner \omega(\psi \models \tau);\ \omega(\psi \models \tau) \lrcorner \mathfrak{C}(\psi \models \tau))))$$

Again, this looming proceeds into a regular parallel process of one-way discourse among several informational entities, expressed by the formula

B20.

$$\delta_r(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta;\ \omega) \equiv$$
$$((\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta) \Vdash$$
$$((\exists(\varphi \in \{\alpha, \beta, \dots, \gamma, \xi, \eta, \dots, \zeta\}).$$
$$(\varphi \Vdash \mathfrak{C}(\varphi);\ \mathfrak{C}(\varphi) \nVdash \varphi;\ \mathfrak{C}(\varphi) \Vdash \omega(\varphi);$$
$$\omega(\varphi) \nVdash \mathfrak{C}(\varphi)));$$
$$(\exists(\psi \in \{\alpha, \beta, \dots, \gamma\}, \tau \in \{\xi, \eta, \dots, \zeta\}).$$
$$((\psi \models \tau) \Vdash \mathfrak{C}(\psi \models \tau);\ \mathfrak{C}(\psi \models \tau) \nVdash (\psi \models \tau);$$
$$\mathfrak{C}(\psi \models \tau) \Vdash \omega(\psi \models \tau);$$
$$\omega(\psi \models \tau) \nVdash \mathfrak{C}(\psi \models \tau)))))$$

It could be said that formula B19 is universalized by replacing $\llcorner$ by $\Vdash$ and $\lrcorner$ by $\nVdash$, getting the equivalent part of B20 or that the equivalent part of B20 is particularized by replacing $\Vdash$ by $\llcorner$ or $\nVdash$ by $\lrcorner$, getting B19. In B20, $\delta_r(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta;\ \omega)$ marks the adequate counter-discourse which continues into discursive embedding.

4.2.3. The Counter-discursive Case $\alpha, \beta \models \alpha, \beta$

In this case, counter-discursive informing takes part between both discursive partners, so that the transmitting and receiving roles of α and β change during the discursive process. It is simply said that between α and β a two-way discourse exists. At the beginning, there is the looming (or bursting) of all possible forms

of counter-informing processes $\mathbb{C}(\alpha)$, $\mathbb{C}(\beta)$, $\mathbb{C}(\alpha \models \beta)$, and $\mathbb{C}(\beta \models \alpha)$ and the corresponding counter-informational products $\omega(\alpha)$, $\omega(\beta)$, $\omega(\alpha \models \beta)$, and $\omega(\beta \models \alpha)$, produced by counter-informing processes in one or another way. Similarly to B15 there is

B21.

$(\alpha, \beta \models \alpha, \beta) \llcorner$
$(\alpha \llcorner \mathbb{C}(\alpha); \mathbb{C}(\alpha) \lrcorner \alpha; \mathbb{C}(\alpha) \llcorner \omega(\alpha); \omega(\alpha) \lrcorner \mathbb{C}(\alpha);$
$\beta \llcorner \mathbb{C}(\beta); \mathbb{C}(\beta) \lrcorner \beta; \mathbb{C}(\beta) \llcorner \omega(\beta); \omega(\beta) \lrcorner \mathbb{C}(\beta);$
$(\alpha \models \beta) \llcorner \mathbb{C}(\alpha \models \beta); \mathbb{C}(\alpha \models \beta) \lrcorner (\alpha \models \beta);$
$\mathbb{C}(\alpha \models \beta) \llcorner \omega(\alpha \models \beta); \omega(\alpha \models \beta) \lrcorner \mathbb{C}(\alpha \models \beta);$
$(\beta \models \alpha) \llcorner \mathbb{C}(\beta \models \alpha); \mathbb{C}(\beta \models \alpha) \lrcorner (\beta \models \alpha);$
$\mathbb{C}(\beta \models \alpha) \llcorner \omega(\beta \models \alpha); \omega(\beta \models \alpha) \lrcorner \mathbb{C}(\beta \models \alpha))$

This formula includes four counter-informational processes for each entity α, β, $\alpha \models \beta$, and $\beta \models \alpha$, respectively, and describes the beginning of the arising of counter-discourses $\delta(\alpha, \omega)$, $\delta(\beta, \omega)$, $\delta(\alpha \models \beta; \omega)$, and $\delta(\beta \models \alpha; \omega)$ out of discourse $\delta(\alpha, \beta \models \alpha, \beta)$. After the occurrence of looming, the looming processes of counter-informing pass over to their regular parallel forms, thus, to the resulting counter-discourse $\delta_r(\alpha, \beta \models \alpha, \beta; \omega)$ within $\alpha, \beta \models \alpha, \beta$:

B22.

$\delta_r(\alpha, \beta \models \alpha, \beta; \omega) \equiv$
$((\alpha, \beta \models \alpha, \beta) \Vdash$
$(\alpha \Vdash \mathbb{C}(\alpha); \mathbb{C}(\alpha) \dashv\Vdash \alpha; \mathbb{C}(\alpha) \Vdash \omega(\alpha); \omega(\alpha) \dashv\Vdash \mathbb{C}(\alpha);$
$\beta \Vdash \mathbb{C}(\beta); \mathbb{C}(\beta) \dashv\Vdash \beta; \mathbb{C}(\beta) \Vdash \omega(\beta); \omega(\beta) \dashv\Vdash \mathbb{C}(\beta);$
$(\alpha \models \beta) \Vdash \mathbb{C}(\alpha \models \beta); \mathbb{C}(\alpha \models \beta) \dashv\Vdash (\alpha \models \beta);$
$\mathbb{C}(\alpha \models \beta) \Vdash \omega(\alpha \models \beta); \omega(\alpha \models \beta) \dashv\Vdash \mathbb{C}(\alpha \models \beta);$
$(\beta \models \alpha) \Vdash \mathbb{C}(\beta \models \alpha); \mathbb{C}(\beta \models \alpha) \dashv\Vdash (\beta \models \alpha);$
$\mathbb{C}(\beta \models \alpha) \Vdash \omega(\beta \models \alpha); \omega(\beta \models \alpha) \dashv\Vdash \mathbb{C}(\beta \models \alpha)))$

So far, this formula completes the discussion concerning the resultant counter-discursive component $\delta_r(\alpha, \beta \models \alpha, \beta; \omega)$ belonging to the process $\alpha, \beta \models \alpha, \beta$.

4.2.3. The Counter-discursive Case $\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$

In this case it is assumed that informational entities $\alpha, \beta, \ldots, \gamma$ participate equally and mutually in the process of discourse. It can be simply said that among $\alpha, \beta, \ldots, \gamma$ a two-way discourse exists. At the beginning, there is the looming (or bursting) of all possible forms of counter-informing processes

B23. $\mathbb{C}(\alpha), \mathbb{C}(\beta), \ldots, \mathbb{C}(\gamma)$ and
$\mathbb{C}(\varphi \models \psi); \varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}$

and the corresponding counter-informational products

B24. $\omega(\alpha), \omega(\beta), \ldots, \omega(\gamma)$ and
$\omega(\varphi \models \psi); \varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}$

produced by counter-informing processes in one or another way. Similarly to B21 there is

B25.

$(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma) \llcorner$
$((\exists(\varphi \in \{\alpha, \beta, \ldots, \gamma\}).$
$(\varphi \llcorner \mathbb{C}(\varphi); \mathbb{C}(\varphi) \lrcorner \varphi; \mathbb{C}(\varphi) \llcorner \omega(\varphi);$
$\omega(\varphi) \lrcorner \mathbb{C}(\varphi)));$
$(\exists((\varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}) \wedge (\varphi \neq \psi)).$
$((\varphi \models \psi) \llcorner \mathbb{C}(\varphi \models \psi); \mathbb{C}(\varphi \models \psi) \lrcorner (\varphi \models \psi);$
$\mathbb{C}(\varphi \models \psi) \llcorner \omega(\varphi \models \psi); \omega(\varphi \models \psi) \lrcorner \mathbb{C}(\varphi \models \psi))))$

This formula includes four counter-informational processes for each informational entity $\alpha, \beta, \ldots, \gamma$ (in fact, self-discursive counter-informational components) and for each interdiscursive process $\varphi \models \psi$, where $\varphi \neq \psi$ and variables φ and ψ fly over entities $\alpha, \beta, \ldots, \gamma$. Thus, this formula describes the beginning of the arising of counter-discourses $\delta(\alpha, \omega)$, $\delta(\beta, \omega)$, $\ldots$, $\delta(\gamma, \omega)$ and $\delta(\varphi \models \psi; \omega)$, where again $\varphi \neq \psi$ and variables φ and ψ fly over entities $\alpha, \beta, \ldots, \gamma$, out of discourse $\delta(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma)$. After the occurrence of looming, the looming processes of counter-informing pass over to their regular parallel forms, thus, to the resulting counter-discourse $\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \omega)$ within $\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$:

B26.

$\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \omega) \equiv$
$((\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma) \llcorner$
$(((\exists(\varphi \in \{\alpha, \beta, \ldots, \gamma\}).$
$(\varphi \Vdash \mathbb{C}(\varphi); \mathbb{C}(\varphi) \dashv\Vdash \varphi; \mathbb{C}(\varphi) \Vdash \omega(\varphi);$
$\omega(\varphi) \dashv\Vdash \mathbb{C}(\varphi)));$
$(\exists((\varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}) \wedge (\varphi \neq \psi)).$
$((\varphi \models \psi) \Vdash \mathbb{C}(\varphi \models \psi); \mathbb{C}(\varphi \models \psi) \dashv\Vdash (\varphi \models \psi);$
$\mathbb{C}(\varphi \models \psi) \Vdash \omega(\varphi \models \psi);$
$\omega(\varphi \models \psi) \dashv\Vdash \mathbb{C}(\varphi \models \psi)))))$

This formula completes the discussion concerning the resultant counter-discursive component $\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \omega)$ belonging to the two-way informational process $\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$.

4.3. Informational Embedding within a Discourse among Several Informational Entities

We have to determine four resulting embedding discourses, namely,

B27. $\delta_r(\alpha \models \beta; \varepsilon)$,
$\delta_r(\alpha, \beta, \ldots, \gamma \models \xi, \eta, \ldots, \zeta; \varepsilon)$,
$\delta_r(\alpha, \beta \models \alpha, \beta; \varepsilon)$, and
$\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \varepsilon)$

The first two cases belong to one-way discourse and the second two cases to two-way discourse. As any information, also these discursive components first loom and then inform out of counter-informational discursive components, thus, having their looming and then their

parallel informing phases. This embedding phenomenology becomes similar to the previous, counter-informational one.

4.3.1. Embedding Discourse within the One-way Process $\alpha \models \beta$

The looming of informational embedding $\mathfrak{E}$ and embedding information ε proceeds out of arisen counter-information ω. Methodologically, in the case of $\alpha \models \beta$, there is $\delta(\alpha \models \beta;\ \omega) \models \delta(\alpha \models \beta;\ \varepsilon)$. As counter-informational discourse, embedding discourse is only a part within the cyclic discursive process of $\alpha \models \beta$. It is possible to construct the following looming process:

B28.

$$\delta_r(\alpha \models \beta;\ \omega) \llcorner$$
$$(\omega(\alpha) \llcorner \mathfrak{E}(\alpha);\ \mathfrak{E}(\alpha) \lrcorner \omega(\alpha);\ \mathfrak{E}(\alpha) \llcorner \varepsilon(\alpha);$$
$$\varepsilon(\alpha) \lrcorner \mathfrak{E}(\alpha);$$
$$\omega(\beta) \llcorner \mathfrak{E}(\beta);\ \mathfrak{E}(\beta) \lrcorner \omega(\beta);\ \mathfrak{E}(\beta) \llcorner \varepsilon(\beta);$$
$$\varepsilon(\beta) \lrcorner \mathfrak{E}(\beta);$$
$$\omega(\alpha \models \beta) \llcorner \mathfrak{E}(\alpha \models \beta);\ \mathfrak{E}(\alpha \models \beta) \lrcorner \omega(\alpha \models \beta);$$
$$\mathfrak{E}(\alpha \models \beta) \llcorner \varepsilon(\alpha \models \beta);\ \varepsilon(\alpha \models \beta) \lrcorner \mathfrak{E}(\alpha \models \beta))$$

The embedding discourse for the case $\alpha \models \beta$ after looming is the following

B29.

$$\delta_r(\alpha \models \beta;\ \varepsilon) \equiv$$
$$(\delta_r(\alpha \models \beta;\ \omega) \Vdash$$
$$(\omega(\alpha) \Vdash \mathfrak{E}(\alpha);\ \mathfrak{E}(\alpha) \nVdash \omega(\alpha);\ \mathfrak{E}(\alpha) \Vdash \varepsilon(\alpha);$$
$$\varepsilon(\alpha) \nVdash \mathfrak{E}(\alpha);$$
$$\omega(\beta) \Vdash \mathfrak{E}(\beta);\ \mathfrak{E}(\beta) \nVdash \omega(\beta);\ \mathfrak{E}(\beta) \Vdash \varepsilon(\beta);$$
$$\varepsilon(\beta) \nVdash \mathfrak{E}(\beta);$$
$$\omega(\alpha \models \beta) \Vdash \mathfrak{E}(\alpha \models \beta);\ \mathfrak{E}(\alpha \models \beta) \nVdash \omega(\alpha \models \beta);$$
$$\mathfrak{E}(\alpha \models \beta) \Vdash \varepsilon(\alpha \models \beta);\ \varepsilon(\alpha \models \beta) \nVdash \mathfrak{E}(\alpha \models \beta)))$$

This formula completes the discussion on one-way embedding discourse of the case $\alpha \models \beta$.

4.3.2. Embedding Discourse within the One-way Process $\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta$

At the beginning of this one-way case of embedding discourse there is the usual looming process:

B30.

$$\delta_r(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta;\ \omega) \llcorner$$
$$((\exists(\varphi \in \{\alpha, \beta, \dots, \gamma, \xi, \eta, \dots, \zeta\}).$$
$$(\omega(\varphi) \llcorner \mathfrak{E}(\varphi);\ \mathfrak{E}(\varphi) \lrcorner \omega(\varphi);\ \mathfrak{E}(\varphi) \llcorner \varepsilon(\varphi);$$
$$\varepsilon(\varphi) \lrcorner \mathfrak{E}(\varphi)));$$
$$(\exists(\psi \in \{\alpha, \beta, \dots, \gamma\},\ \tau \in \{\xi, \eta, \dots, \zeta\}).$$
$$(\omega(\psi \models \tau) \llcorner \mathfrak{E}(\psi \models \tau);\ \mathfrak{E}(\psi \models \tau) \lrcorner \omega(\psi \models \tau);$$
$$\mathfrak{E}(\psi \models \tau) \llcorner \varepsilon(\psi \models \tau);\ \varepsilon(\psi \models \tau) \lrcorner \mathfrak{E}(\psi \models \tau))))$$

This looming proceeds into a regular parallel process of one-way embedding discourse, marked by $\delta_r(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta;\ \varepsilon)$, among several informational entities, and can be expressed by the formula

B31.

$$\delta_r(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta;\ \varepsilon) \equiv$$
$$(\delta_r(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta;\ \omega) \Vdash$$
$$((\exists(\varphi \in \{\alpha, \beta, \dots, \gamma, \xi, \eta, \dots, \zeta\}).$$
$$(\omega(\varphi) \Vdash \mathfrak{E}(\varphi);\ \mathfrak{E}(\varphi) \nVdash \omega(\varphi);\ \mathfrak{E}(\varphi) \Vdash \varepsilon(\varphi);$$
$$\varepsilon(\varphi) \nVdash \mathfrak{E}(\varphi)));$$
$$(\exists(\psi \in \{\alpha, \beta, \dots, \gamma\},\ \tau \in \{\xi, \eta, \dots, \zeta\}).$$
$$(\omega(\psi \models \tau) \Vdash \mathfrak{E}(\psi \models \tau);\ \mathfrak{E}(\psi \models \tau) \nVdash \omega(\psi \models \tau);$$
$$\mathfrak{E}(\psi \models \tau) \Vdash \varepsilon(\psi \models \tau);$$
$$\varepsilon(\psi \models \tau) \nVdash \mathfrak{E}(\psi \models \tau)))))$$

This formula completes the one-way case of embedding discourse $\delta_r(\alpha, \beta, \dots, \gamma \models \xi, \eta, \dots, \zeta;\ \varepsilon)$.

4.3.3. Embedding Discourse within the Two-way Process $\alpha, \beta \models \alpha, \beta$

In this case, embedding-discursive informing takes part between both discursive partners α and β, so that the transmitting and receiving roles of α and β change during the discursive process. At the beginning, there is the looming of all possible forms of embedding processes $\mathfrak{E}(\alpha)$, $\mathfrak{E}(\beta)$, $\mathfrak{E}(\alpha \models \beta)$, and $\mathfrak{E}(\beta \models \alpha)$ and the corresponding embedding products $\varepsilon(\alpha)$, $\varepsilon(\beta)$, $\varepsilon(\alpha \models \beta)$, and $\varepsilon(\beta \models \alpha)$, produced by embedding processes in one or another way. The looming of the discursive embedding phenomenon is the following:

B32.

$$\delta_r(\alpha, \beta \models \alpha, \beta;\ \omega) \llcorner$$
$$(\omega(\alpha) \llcorner \mathfrak{E}(\alpha);\ \mathfrak{E}(\alpha) \lrcorner \omega(\alpha);\ \mathfrak{E}(\alpha) \llcorner \varepsilon(\alpha);$$
$$\varepsilon(\alpha) \lrcorner \mathfrak{E}(\alpha);$$
$$\omega(\beta) \llcorner \mathfrak{E}(\beta);\ \mathfrak{E}(\beta) \lrcorner \omega(\beta);\ \mathfrak{E}(\beta) \llcorner \varepsilon(\beta);$$
$$\varepsilon(\beta) \lrcorner \mathfrak{E}(\beta);$$
$$\omega(\alpha \models \beta) \llcorner \mathfrak{E}(\alpha \models \beta);\ \mathfrak{E}(\alpha \models \beta) \lrcorner \omega(\alpha \models \beta);$$
$$\mathfrak{E}(\alpha \models \beta) \llcorner \varepsilon(\alpha \models \beta);\ \varepsilon(\alpha \models \beta) \lrcorner \mathfrak{E}(\alpha \models \beta);$$
$$\omega(\beta \models \alpha) \llcorner \mathfrak{E}(\beta \models \alpha);\ \mathfrak{E}(\beta \models \alpha) \lrcorner \omega(\beta \models \alpha);$$
$$\mathfrak{E}(\beta \models \alpha) \llcorner \varepsilon(\beta \models \alpha);\ \varepsilon(\beta \models \alpha) \lrcorner \mathfrak{E}(\beta \models \alpha))$$

This formula includes four informationally embedding processes for each entity α, β, $\alpha \models \beta$, and $\beta \models \alpha$, respectively, and describes the beginning of the arising of embedding discourses $\delta(\alpha, \varepsilon)$, $\delta(\beta, \varepsilon)$, $\delta(\alpha \models \beta, \varepsilon)$, and $\delta(\beta \models \alpha;\ \varepsilon)$ out of discourse $\delta(\alpha, \beta \models \alpha, \beta;\ \gamma)$. After the occurrence of looming, the looming processes of embedding pass over to their regular parallel forms, thus, to the resulting discourse of embedding $\delta_r(\alpha, \beta \models \alpha, \beta;\ \varepsilon)$ within $\alpha, \beta \models \alpha, \beta$:

B33.

$$\delta_r(\alpha, \beta \models \alpha, \beta;\ \varepsilon) \equiv$$
$$(\delta_r(\alpha, \beta \models \alpha, \beta;\ \omega) \Vdash$$
$$(\omega(\alpha) \Vdash \mathfrak{E}(\alpha);\ \mathfrak{E}(\alpha) \nVdash \omega(\alpha);\ \mathfrak{E}(\alpha) \Vdash \varepsilon(\alpha);$$
$$\varepsilon(\alpha) \nVdash \mathfrak{E}(\alpha);$$

$\omega(\beta) \models \mathfrak{E}(\beta)$; $\mathfrak{E}(\beta) \not\models \omega(\beta)$; $\mathfrak{E}(\beta) \models \varepsilon(\beta)$;
$\varepsilon(\beta) \not\models \mathfrak{E}(\beta)$;
$\omega(\alpha \models \beta) \models \mathfrak{E}(\alpha \models \beta)$; $\mathfrak{E}(\alpha \models \beta) \not\models \omega(\alpha \models \beta)$;
$\mathfrak{E}(\alpha \models \beta) \models \varepsilon(\alpha \models \beta)$; $\varepsilon(\alpha \models \beta) \not\models \mathfrak{E}(\alpha \models \beta)$;
$\omega(\beta \models \alpha) \models \mathfrak{E}(\beta \models \alpha)$; $\mathfrak{E}(\beta \models \alpha) \not\models \omega(\beta \models \alpha)$;
$\mathfrak{E}(\beta \models \alpha) \models \varepsilon(\beta \models \alpha)$; $\varepsilon(\beta \models \alpha) \not\models \mathfrak{E}(\beta \models \alpha)$)))

So far, this formula completes the discussion concerning the resultant embedding component $\delta_r(\alpha, \beta \models \alpha, \beta; \varepsilon)$ belonging to the process $\alpha, \beta \models \alpha, \beta$.

4.3.4. Embedding Discourse within the Two-way Process $\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$

In this case it is assumed that informational entities $\alpha, \beta, \ldots, \gamma$ participate equally and mutually in the embedding part of discourse. It can be simply said that among $\alpha, \beta, \ldots, \gamma$ a two-way embedding discourse exists. At the beginning, there is the looming (or bursting) of all possible forms of informationally embedding processes

B34. $\mathfrak{E}(\varphi \models \psi)$; $\varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}$,
where $\mathfrak{E}(\varphi \models \varphi) \equiv \mathfrak{E}(\varphi)$

and the corresponding products of embedding information

B35. $\varepsilon(\varphi \models \psi)$; $\varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}$,
where $\varepsilon(\varphi \models \varphi) \equiv \varepsilon(\varphi)$

produced by embedding processes in one or another way. The looming of embedding discourse within this case can be expressed by

B36.
$\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \omega) \sqsubset$
$((\exists(\varphi \in \{\alpha, \beta, \ldots, \gamma\}))\bullet$
$(\omega(\varphi) \sqsubset \mathfrak{E}(\varphi)$; $\mathfrak{E}(\varphi) \sqcup \omega(\varphi)$; $\mathfrak{E}(\varphi) \sqsubset \varepsilon(\varphi)$;
$\varepsilon(\varphi) \sqcup \mathfrak{E}(\varphi)))$;
$(\exists((\varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}) \wedge (\varphi \neq \psi)))\bullet$
$(\omega(\varphi \models \psi) \sqsubset \mathfrak{E}(\varphi \models \psi)$; $\mathfrak{E}(\varphi \models \psi) \sqcup \omega(\varphi \models \psi)$;
$\mathfrak{E}(\varphi \models \psi) \sqsubset \varepsilon(\varphi \models \psi)$; $\varepsilon(\varphi \models \psi) \sqcup \mathfrak{E}(\varphi \models \psi))))$

This formula includes four embedding-informational processes for each informational entity $\alpha, \beta, \ldots, \gamma$ (in fact, self-discursive embedding-informational components) and for each interdiscursive process $\varphi \models \psi$, where $\varphi \neq \psi$ and variables φ and ψ fly over entities $\alpha, \beta, \ldots, \gamma$. Thus, this formula describes the beginning of the arising of the embedding discourses $\delta(\alpha, \varepsilon)$, $\delta(\beta, \varepsilon)$, ... , $\delta(\gamma, \varepsilon)$ and $\delta(\varphi \models \psi; \varepsilon)$, where again $\varphi \neq \psi$ and variables φ and ψ fly over entities $\alpha, \beta, \ldots, \gamma$, out of counter-discourse $\delta(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \omega)$. After the occurrence of looming, the looming processes of embedding pass over to their regular parallel forms, thus, to the resulting discourse of embedding $\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \varepsilon)$ within $\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$:

B37.
$\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \varepsilon) \equiv$
$(\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \omega) \sqsubset$
$(((\exists(\varphi \in \{\alpha, \beta, \ldots, \gamma\}))\bullet$
$(\omega(\varphi) \models \mathfrak{E}(\varphi)$; $\mathfrak{E}(\varphi) \not\models \omega(\varphi)$; $\mathfrak{E}(\varphi) \models \varepsilon(\varphi)$;
$\varepsilon(\varphi) \not\models \mathfrak{E}(\varphi)))$;
$(\exists((\varphi, \psi \in \{\alpha, \beta, \ldots, \gamma\}) \wedge (\varphi \neq \psi)))\bullet$
$(\omega(\varphi \models \psi) \models \mathfrak{E}(\varphi \models \psi)$; $\mathfrak{E}(\varphi \models \psi) \not\models \omega(\varphi \models \psi)$;
$\mathfrak{E}(\varphi \models \psi) \models \varepsilon(\varphi \models \psi)$;
$\varepsilon(\varphi \models \psi) \not\models \mathfrak{E}(\varphi \models \psi)))))$

This formula completes the discussion concerning the resultant embedding-informational component $\delta_r(\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma; \varepsilon)$ belonging to the two-way informational process of the form $\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$.

4.4. The Game of Counter-informing and Informational Embedding within a Discourse among Several Informational Entities

The course of discourse among various informational entities depends essentially on the game in which informing (initial or original discourse), counter-informing (counter-discourse), and informational embedding (embedding-discourse) take part as substantial informational players within discursive nature of information. This game is circular in the sense that after the looming of resulting discourse $\delta_r(\alpha)$, this is closed via resulting counter-discourse $\delta_r(\alpha, \omega)$ and resulting embedding discourse $\delta_r(\alpha, \varepsilon)$ into the so-called regular (parallel, cyclic) discursive cycle, i.e. looming back into the original discourse $\delta_r(\alpha)$. This cycle was dynamically schematized by formula DCω in section 3.3.

In regard to our four cases we have according to DCω

B38. $((\delta_r(\alpha) \models \delta_r(\alpha, \omega)) \models \delta_r(\alpha, \varepsilon)) \models \delta_r(\alpha)$,
where α stands for
$\alpha \models \beta$;
$\alpha, \beta, \ldots, \gamma \models \xi, \eta, \ldots, \zeta$;
$\alpha, \beta \models \alpha, \beta$; or
$\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$

The game of discursive looming as the beginning of the game of discourse can be described according to LDα in section 3.3:

B39.
$((\alpha \sqsubset$
$(\alpha \sqsubset \mathfrak{C}(\alpha)$; $\mathfrak{C}(\alpha) \sqcup \alpha$; $\mathfrak{C}(\alpha) \sqsubset \omega(\alpha)$;
$\omega(\alpha) \sqcup \mathfrak{C}(\alpha))) \sqsubset$
$(\omega(\alpha) \sqsubset \mathfrak{E}(\alpha)$; $\mathfrak{E}(\alpha) \sqcup \omega(\alpha)$; $\mathfrak{E}(\alpha) \sqsubset \varepsilon(\alpha$;
$\varepsilon \sqcup \mathfrak{E})) \sqsubset$
$(\varepsilon(\alpha) \sqsubset \alpha$; $\alpha \sqcup \varepsilon(\alpha))$,
where α stands for

$\alpha \models \beta$;
$\alpha, \beta, \ldots, \gamma \models \xi, \eta, \ldots, \zeta$;
$\alpha, \beta \models \alpha, \beta$; or
$\alpha, \beta, \ldots, \gamma \models \alpha, \beta, \ldots, \gamma$

We see how α looms the entire, cyclic discursive process within itself, since at the end of the last formula the embedding information $\varepsilon(\alpha)$ looms back into α.

After looming, formula B38 describes a regular discursive process within α, where discursive components appearing in B38 are the following:

B40. $\delta_r(\alpha) \equiv (\alpha \models \alpha)$;
$\delta_r(\alpha, \omega) \equiv (\alpha \models \mathbb{C}(\alpha);\ \mathbb{C}(\alpha) \dashv \alpha;$
$\mathbb{C}(\alpha) \models \omega(\alpha);\ \omega(\alpha) \dashv \mathbb{C}(\alpha))$;
$\delta_r(\alpha, \varepsilon) \equiv (\omega(\alpha) \models \mathbb{E}(\alpha);\ \mathbb{E}(\alpha) \dashv \omega(\alpha);$
$\mathbb{E}(\alpha) \models \varepsilon(\alpha);\ \varepsilon(\alpha) \dashv \mathbb{E}(\alpha))$;
$\delta_r(\alpha) \equiv (\varepsilon(\alpha) \models \alpha;\ \alpha \dashv \varepsilon(\alpha))$

The first and the fourth equivalence are in no way in contradiction, since, by definition, $\varepsilon(\alpha)$ is an internal affair of α. Thus, also

B41. $\delta_r(\alpha) \equiv (\alpha, \varepsilon \models \alpha;\ \alpha \dashv \alpha, \varepsilon)$

reflects the known phenomenology of an arbitrary informational entity α. By this kind of discussion, phenomena of the discursive nature of information, considering specific discursive components, are believed to be sufficiently clarified.

5. LACANIAN FORMS OF DISCOURSE

> ... The false as well as true science can be put into formulas.
>
> Jacques Lacan [9] 17

5.1. A General Scenario of Lacanian Discourse

> ... Nature provides us with, let us speak out also this word, markers and these markers organize in an inaugural manner human relations, give them structures and model them.
>
> Jacques Lacan [9] 26

The ideas of treating the so-called Lacanian discourse in the way of informational logic have been mainly seized from Bracher [7]. Later, in the course of informational analysis, it could be demonstrated that the apparatus of informational logic enables analysis, which might go behind the Lacanian ideas, more and more into informational details, bringing to the surface constructive capabilities of the Lacanian concept of discourse.

A discourse as informational process (in brain, within interaction of the living) produces informational effects in psychical economies of relative informational transmitter α and relative informational receptor β, i.e. in the metaphysical or informationally total domain of α and β. In general, both α and β can be understood as autopoietical informational phenomenon being involved or mutually and individually impacted by the process of discourse. It is possible to imagine how a two-way discursive process, symbolically expressed by $\alpha, \beta \models \alpha, \beta$, changes the social behavior and how it is informationally thrown into the domain of wish rather than into the domain of knowledge. This conclusion might not be important on the general level of discussion, however, can become relevant at the detailed analysis of discursive phenomenology of information.

It is possible to think that information, which informs, interpellates information which is addressed by informing and that this informational interpellation is a specific or particular function or operation marked by the discursive metaoperator $\models$ occurring between impacting and impacted informational entities α and β. The two-way communicational process marked by $\alpha, \beta \models \alpha, \beta$ performs (or informs) a specific (or particular) type of information (or informational arising), within which the relative roles of transmitters and receptors are exchanged during the flow of discourse.

According to Lacan, it is possible to study (or introduce) four basic entities, called performing (acting, behaving), truth, Other and production and mark them symbolically by ϑ, τ, ρ, and λ, respectively. Further, it is possible to decompose the circumstantial (relative) transmitter α into a self-discursive process of the form $(\vartheta_\alpha \models \tau_\alpha) \models (\rho_\alpha \models \lambda_\alpha)$ or specifically (Lacanianly) into the form $(\vartheta_\alpha / \tau_\alpha) \models (\rho_\alpha / \lambda_\alpha)$; similarly, the circumstantial (relative) receptor β can be decomposed into $(\vartheta_\beta \models \tau_\beta) \models (\rho_\beta \models \lambda_\beta)$ or specifically (Lacanianly) into the form $(\vartheta_\beta / \tau_\beta) \models (\rho_\beta / \lambda_\beta)$; further, the discursive process between transmitter α and receptor β, i.e. $\alpha \models \beta$, can be decomposed into $(\vartheta_\alpha \models \tau_\alpha) \models (\rho_\beta \models \lambda_\beta)$ or specifically (Lacanianly) into the form $(\vartheta_\alpha / \tau_\alpha) \models (\rho_\beta / \lambda_\beta)$; finally, the discursive process between receptor β and transmitter α, i.e. $\beta \models \alpha$, can be decomposed into the interdiscursive process of the form $(\vartheta_\beta \models \tau_\beta) \models (\rho_\alpha \models \lambda_\alpha)$ or specifically (Lacanianly) into the form $(\vartheta_\beta / \tau_\beta) \models (\rho_\alpha / \lambda_\alpha)$. In these expressions, "/" and "$\models$" are particular (Lacanian) informational operators. The general form of these processes occurring within circumstantial (relative) transmitter α, circumstantial (relative) receptor β, and between circumstantial (relative) transmitter α and circumstantial (relative) receptor β, and vice versa, can be decomposed as

L1. $(\alpha, \beta \models \alpha, \beta) \models$
$((\vartheta_\alpha \models \tau_\alpha), (\vartheta_\beta \models \tau_\beta) \models$
$(\rho_\alpha \models \lambda_\alpha), (\rho_\beta \models \lambda_\beta))$

or specifically (Lacanianly)

L2. $(\alpha, \beta \models \alpha, \beta) \models$
$((\vartheta_\alpha / \tau_\alpha), (\vartheta_\beta / \tau_\beta) \models$
$(\rho_\alpha / \lambda_\alpha), (\rho_\beta / \lambda_\beta))$

It is worth to mention the following important facts to these formulas: formula $\alpha, \beta \models \alpha, \beta$ in L1 and L2 ensures all possible cases of self and mutual discourse concerning relative transmitter α and relative receptor β, i.e. the processes $\alpha \models \alpha$, $\alpha \models \beta$, $\beta \models \alpha$, and $\beta \models \beta$. Further, operators $\models$ and / appearing in L1 and L2 can be particularized to some general degree, for instance, in the case of L2 into

L2'. $(\alpha, \beta \models \alpha, \beta) \models$
$((\vartheta_\alpha /_\alpha \tau_\alpha), (\vartheta_\beta /_\beta \tau_\beta) \models_\gamma$
$(\rho_\alpha /_\alpha \lambda_\alpha), (\rho_\beta /_\beta \lambda_\beta))$

We see, for instance, that for the discourse within the transmitter α (self-informational form of discourse) only the transmitter is impacting the operator "/", and similar is valid for the receptor β. In the case of two-way or interinformational discourse between α and β, both entities impact the operator $\models$, so $\models_\gamma$.

However, Lacan extracts (for instance, by modus ponens [6]) the informational discursive components ϑ / τ and ρ / λ out of transmitter and receptor information and connects them discursively, postulating

L3. $(\vartheta_\alpha / \tau_\alpha) \models (\rho_\beta / \lambda_\beta);$
$(\vartheta_\beta / \tau_\beta) \models (\rho_\alpha / \lambda_\alpha)$

since roles of transmitter and receptor within a developing discourse can be changed whensoever. This scheme is the basic origin (or syntactic background) of any Lacanian type of discourse. Onto this scheme (or informational formula) various particularizations of operands and operators can be rotated. Thus, L3 should be the basic model of social interaction and communication where the left part $(\vartheta_\alpha / \tau_\alpha)$ is occupied by transmitting and the right part $(\rho_\beta / \lambda_\beta)$ by receiving information, when α transmits information and β receives it, and vice versa, when the roles of α and β are changed. This changing of roles happens frequently through the course of discourse.

It is quite believable that Lacan has considered the so-called self-discursive processes within the transmitter and receptor. In our case, L3 can be completed by the systematic extraction (modus ponens) concerning L2:

L3'. $(\vartheta_\alpha / \tau_\alpha) \models (\rho_\alpha / \lambda_\alpha);$
$(\vartheta_\alpha / \tau_\alpha) \models (\rho_\beta / \lambda_\beta);$
$(\vartheta_\beta / \tau_\beta) \models (\rho_\alpha / \lambda_\alpha);$
$(\vartheta_\beta / \tau_\beta) \models (\rho_\beta / \lambda_\beta)$

The last scheme determines the Lacanian form of discourse in several details and can be understood as the decomposition of the basic Lacanian scheme of discourse, i.e.,

L3". $(\vartheta / \tau) \models (\rho / \lambda)$

for the case of two participants α and β in the (two-way) process of discourse.

Some additional comments might be useful for the understanding of Lacanian discourse. The performing (or acting) ϑ (the place it occupies in the formula) always means also informational domination within a discourse. This place is occupied by the factor (or factorial information) which dominates (informs) as the speaker (transmitter) or as the expression of the writer (within metaphysics of discursively involved living agents). The performing ϑ supports the truth τ which is the condition of possibility π for ϑ. It is to understand that possibility π is constituted metaphysically and environmentally through discursive partners, for instance, α and β, thus $\alpha \models \pi_\alpha$ and $\beta \models \pi_\beta$, respectively. For instance, the following, senseful decomposition is possible:

L4. $(\vartheta \models \tau; \tau \models \pi; \pi \models \vartheta) \models$
$((\vartheta_\alpha \models \tau_\alpha; \tau_\alpha \models \pi_\alpha; \pi_\alpha \models \vartheta_\alpha);$
$(\vartheta_\beta \models \tau_\beta; \tau_\beta \models \pi_\beta; \pi_\beta \models \vartheta_\beta))$

Thus, the transmitter and receptor information α and β include ($\subset$) the discursively relevant components ϑ, τ, (ϑ / τ), and π, i.e.,

L5. $\vartheta_\alpha, \tau_\alpha, (\vartheta_\alpha / \tau_\alpha), \pi_\alpha \subset \alpha;$
$\vartheta_\beta, \tau_\beta, (\vartheta_\beta / \tau_\beta), \pi_\beta \subset \beta$

and these components inform, i.e., $\vartheta_\alpha \models$, $\tau_\alpha \models$, $(\vartheta_\alpha / \tau_\alpha) \models$, $\pi_\alpha \models$, $\vartheta_\beta \models$, $\tau_\beta \models$, $(\vartheta_\beta / \tau_\beta) \models$, $\pi_\beta \models$, and are informed, i.e., $\models \vartheta_\alpha$, $\models \tau_\alpha$, $\models (\vartheta_\alpha / \tau_\alpha)$, $\models \pi_\alpha$, $\models \vartheta_\beta$, $\models \tau_\beta$, $\models (\vartheta_\beta / \tau_\beta)$, $\models \pi_\beta$ (at least by itself and probably by other information). Both ϑ and τ inform cyclically via information π, i.e.

L6. $\alpha \models ((((\vartheta_\alpha \models \tau_\alpha) \models \pi_\alpha) \models \vartheta_\alpha);$
$(((\tau_\alpha \models \pi_\alpha) \models \vartheta_\alpha) \models \tau_\alpha));$
$\beta \models ((((\vartheta_\beta \models \tau_\beta) \models \pi_\alpha) \models \vartheta_\beta);$
$(((\tau_\beta \models \pi_\alpha) \models \vartheta_\beta) \models \tau_\beta));$

Shortly, the transmitting component ϑ / τ is complexly informed, i.e. $\models (\vartheta / \tau)$, and performs (informs) regularly in informational sense.

On the right side of the discursive relations in L3 the receptor components called Other/production, ρ / λ, appeared. Within this construction, ρ is the receptor of the message ϑ / τ. By usual terms, ϑ / τ is called speech. The production λ is information produced by the receptor as the answer to the message. Thus, the process ρ / λ has also the meaning "the Other ρ informs (or informationally coproduces) λ, or generally, as treated in L1, $\rho_\alpha \models \lambda_\alpha$ and $\rho_\beta \models \lambda_\beta$.

Let us now resume the following: in the Lacanian scheme of discourse we have the relative transmitter α and the relative receptor β. In this scheme, α communicates to β by messaging through ϑ / τ, within which the informationally dominating component ϑ masters

or determines the truth τ. On the receptor side β this messaging is specifically accepted through the receiving component ρ / λ, where ρ is the substantial metaphysical component of the receptor β, called the Other, which, within the entire metaphysical domain of β, produces information λ as the consequence or answering to informing of ϑ / τ. Now we see how Lacanian scheme of discourse despite of its initial schematic simplicity becomes more and more informationally complex and begins to expand over its initially simplicistic philosophy. We see how the initial schematic system arises and becomes as complex as we are able to determine (decompose) new and new components and their impacting and impactedness within the arising discursive system. The joint Lacanian transmitting and receiving discursive system can to this point be expressed formally for the case of one-way communication, considering Lacanian postulates L1, ... , L6 and some comments in the following way:

JL→. $\alpha \Vdash \beta$;

LTx→. $\alpha \Vdash (\vartheta_\alpha \,/\, \tau_\alpha)$; $[\alpha \models \alpha, \beta]$
$(\vartheta_\alpha \,/\, \tau_\alpha) \Vdash (\rho_\alpha \,/\, \lambda_\alpha), (\rho_\beta \,/\, \lambda_\beta)$;
$(\rho_\alpha \,/\, \lambda_\alpha) \models \alpha$;
$\alpha \Vdash \pi_\alpha$;
$((\vartheta_\alpha \Vdash \tau_\alpha) \models \pi_\alpha) \Vdash \vartheta_\alpha$;
$((\tau_\alpha \Vdash \pi_\alpha) \models \vartheta_\alpha) \Vdash \tau_\alpha$;

LRx→. $\beta \Vdash (\vartheta_\beta \,/\, \tau_\beta)$; $[\beta \models \beta]$
$(\vartheta_\beta \,/\, \tau_\beta) \Vdash (\rho_\beta \,/\, \lambda_\beta)$;
$(\rho_\beta \,/\, \lambda_\beta) \models \beta$;
$\beta \Vdash \pi_\beta$;

L7. $((\vartheta_\beta \Vdash \tau_\beta) \models \pi_\beta) \Vdash \vartheta_\beta$;
$((\tau_\beta \Vdash \pi_\beta) \models \vartheta_\beta) \Vdash \tau_\beta$

This system can be seen as a minimally particularized one, so, it can be expanded (or decomposed) easily into greater detail. It is to say that also some previous, informationally general concepts of discursive counter-informing and embedding can be considered, e.g., subsumed and/or superscribed to the system already developed. In this system, expressions [...] are comments. Subsystem L7 marks the discursive interaction within the receptor. Within system JL→, β performs as a steady receptor, which does not interact backwards to the transmitter α. Further, operator $\models$ was particularized by operator $\Vdash$, which explicates the parallel processing between transmitter α and receptor β and inside of them. It is also to understand that ϑ_α / τ_α and ϑ_β / τ_β are the so-called speaking parts, and ρ_α / λ_α and ρ_β / λ_β are the listening parts of transmitter and receptor, respectively.

The next question which has to be touched is how can the impacting of receptor on transmitter be brought into consideration. In a real discourse, a two-way interaction comes always into existence, thus the following Lacanian discursive system can be appropriated:

JL↔. $\alpha, \beta \Vdash \alpha, \beta$;

LTx↔. $\alpha, \beta \Vdash (\vartheta_\alpha \,/\, \tau_\alpha), (\vartheta_\beta \,/\, \tau_\beta)$;
$(\vartheta_\alpha \,/\, \tau_\alpha) \Vdash (\rho_\alpha \,/\, \lambda_\alpha), (\rho_\beta \,/\, \lambda_\beta)$;
$(\rho_\alpha \,/\, \lambda_\alpha) \models \alpha$;
$\alpha \Vdash \pi_\alpha$;
$((\vartheta_\alpha \Vdash \tau_\alpha) \models \pi_\alpha) \Vdash \vartheta_\alpha$;
$((\tau_\alpha \Vdash \pi_\alpha) \models \vartheta_\alpha) \Vdash \tau_\alpha$;

LRx↔. $\beta, \alpha \Vdash (\rho_\beta \,/\, \lambda_\beta), (\rho_\alpha \,/\, \lambda_\alpha)$;
$(\vartheta_\beta \,/\, \tau_\beta) \Vdash (\rho_\beta \,/\, \lambda_\beta), (\rho_\alpha \,/\, \lambda_\alpha)$;
$\beta \Vdash \pi_\beta$;
$(\rho_\beta \,/\, \lambda_\beta) \models \beta$;
$((\vartheta_\beta \Vdash \tau_\beta) \models \pi_\beta) \Vdash \vartheta_\beta$;
$((\tau_\beta \Vdash \pi_\beta) \models \vartheta_\beta) \Vdash \tau_\beta$

This system is formally symmetric in regard to the transmitter α and receptor β. It is in no way closed, so it can be always developed (progressively decomposed) to the needed details by adding new formulas and decomposing the appearing operands and operators, and also particularizing and universalizing them. We can see how initial Lacanian idea of discourse becomes more and more formally complex and that this complexity grows with the number of participants in the discourse.

In this way it is possible to show a sufficiently clean Lacanian discursive system of several participants in which each participant is performing harmonically as transmitter and receiver. This completely symmetric system of several participants in a discourse has the form:

L↔. $\alpha, \beta, \ldots, \gamma \Vdash \alpha, \beta, \ldots, \gamma$;

Lα↔. $\alpha \Vdash (\vartheta_\alpha \,/\, \tau_\alpha), (\vartheta_\beta \,/\, \tau_\beta), \ldots, (\vartheta_\gamma \,/\, \tau_\gamma)$;
$(\vartheta_\alpha \,/\, \tau_\alpha) \Vdash (\rho_\alpha \,/\, \lambda_\alpha), (\rho_\beta \,/\, \lambda_\beta), \ldots, (\rho_\gamma \,/\, \lambda_\gamma)$;
$(\rho_\alpha \,/\, \lambda_\alpha) \models \alpha$;
$\alpha \Vdash \pi_\alpha$;
$((\vartheta_\alpha \Vdash \tau_\alpha) \models \pi_\alpha) \Vdash \vartheta_\alpha$;
$((\tau_\alpha \Vdash \pi_\alpha) \models \vartheta_\alpha) \Vdash \tau_\alpha$;
$\alpha \Vdash (\rho_\alpha \,/\, \lambda_\alpha), (\rho_\beta \,/\, \lambda_\beta), \ldots, (\rho_\gamma \,/\, \lambda_\gamma)$;

Lβ↔. $\beta \Vdash (\vartheta_\alpha \,/\, \tau_\alpha), (\vartheta_\beta \,/\, \tau_\beta), \ldots, (\vartheta_\gamma \,/\, \tau_\gamma)$;
$(\vartheta_\beta \,/\, \tau_\beta) \Vdash (\rho_\alpha \,/\, \lambda_\alpha), (\rho_\beta \,/\, \lambda_\beta), \ldots, (\rho_\gamma \,/\, \lambda_\gamma)$;
$(\rho_\beta \,/\, \lambda_\beta) \models \beta$;
$\beta \Vdash \pi_\beta$;
$((\vartheta_\beta \Vdash \tau_\beta) \models \pi_\beta) \Vdash \vartheta_\beta$;
$((\tau_\beta \Vdash \pi_\beta) \models \vartheta_\beta) \Vdash \tau_\beta$
$\beta \Vdash (\rho_\alpha \,/\, \lambda_\alpha), (\rho_\beta \,/\, \lambda_\beta), \ldots, (\rho_\gamma \,/\, \lambda_\gamma)$;

... .

Lγ↔. $\gamma \Vdash (\vartheta_\alpha \,/\, \tau_\alpha), (\vartheta_\beta \,/\, \tau_\beta), \ldots, (\vartheta_\gamma \,/\, \tau_\gamma)$;
$(\vartheta_\gamma \,/\, \tau_\gamma) \Vdash (\rho_\alpha \,/\, \lambda_\alpha), (\rho_\beta \,/\, \lambda_\beta), \ldots, (\rho_\gamma \,/\, \lambda_\gamma)$;
$(\rho_\gamma \,/\, \lambda_\gamma) \models \gamma$;

$$\gamma \models \pi_\gamma;$$
$$((\vartheta_\gamma \models \tau_\gamma) \models \pi_\gamma) \models \vartheta_\gamma;$$
$$((\tau_\gamma \models \pi_\gamma) \models \vartheta_\gamma) \models \tau_\gamma$$
$$\gamma \models (\rho_\alpha / \lambda_\alpha), (\rho_\beta / \lambda_\beta), \ldots, (\rho_\gamma / \lambda_\gamma);$$

This system can be still particularized, universalized, and decomposed according to the arising needs and various philosophies and constructions in accordance with the Lacanian (or psychoanalytic) style (or doctrine) of discourse, however also outside of Lacanian (or psychoanalytic) concepts. As one can observe, there is a slight conceptual difference between informational systems marked by JL↔ and L↔; the reader will be able to discover it by himself or herself.

5.2. On the Notion of the Other as Information

The notion of the Other concerns counter-information. If one says that there is no the Other of the Other, this would mean that there is no counter-information of counter-information. This seems reasonable because counter-information as phenomenology of information is not yet embedded into the so-called comprehension of existing or source information which produces (generates) counter-information. In this respect, it is not possible to distinguish counter-information from counter-information, although counter-information, if marked as such, is nothing other than information. This discussion merely concerns a part of Lacan's hypothesis by which he argues that there does not exist the Other of the Other [8, page 50].

By informational terms, the psychoanalytic term the Other is counter-informational on different levels of discourse. And as we have seen, within each simple or composed informational entity, always an inner discourse, the so-called self-discourse occurs. The Other may appear explicitly in the domain of the so-called counter-discourse and implicitly in any other discursive component as a distributed informational phenomenon within information. To which extent the Other will be brought into the "awareness" of information depends exclusively on informational capability concerning discursive embedding, by which parts of counter-discourse can be embedded into existing discourse and other counter-informational parts can be lost (for ever).

5.3. The Lacan's Idea of the Basic Scheme Appropriation

> ... - the unconscious is structured as language - ... this is linguistics, which model is an operator game performed within its spontaneity completely by itself - precisely this structure delivers the status of unconscious. It confirms that under the notion of unconscious there exists something which can be marked, attained, and objectified.
>
> Jacques Lacan [9] 26, 27

How can the basic Lacanian scheme of discourse L3 be appropriated? If we take this scheme

$$(\vartheta / \tau) \models (\rho / \lambda)$$

then each element (operand or operator) of it can be occupied (appropriated, informationally substituted) by a particular Lacanian entity. In fact, Lacan chooses a cyclic scheme of four operand elements, namely σ_2 marking the knowledge, σ_1 denoting the marker-master, $\mathfrak{A}$ marking the object $\mathfrak{A}$ (plus-de-jouir, also exceeded or remained pleasure), and Φ denoting the split (castrated) subject. To remember this scheme of operand elements it is convenient to put them into the Lacanian matrix form

L8.

$$\begin{matrix} \sigma_2 & \mathfrak{A} \\ \sigma_1 & \Phi \end{matrix}$$

so that this matrix can be rotated clockwise, giving four possible matrix types, i.e.

L9.

$$\begin{matrix} \sigma_2 & \mathfrak{A} \\ \sigma_1 & \Phi \end{matrix} \qquad \begin{matrix} \sigma_1 & \sigma_2 \\ \Phi & \mathfrak{A} \end{matrix} \qquad \begin{matrix} \Phi & \sigma_1 \\ \mathfrak{A} & \sigma_2 \end{matrix} \qquad \begin{matrix} \mathfrak{A} & \Phi \\ \sigma_2 & \sigma_1 \end{matrix}$$

which will be characteristic for the so-called university, master's, hysteric's, and analyst's discourse, respectively.

The question to be cleared concerns the possible meanings of discursive entities σ_2, σ_1, $\mathfrak{A}$, and Φ and their informationally circular impacting. These entities can be understood as informational processes which roughly mark the knowledge (e.g. cognition, belief, faith), marker-master (e.g. truth, ideal, ideology), remnant (Lacanian object, marked by "a"), and split subject (as far as it is constructed as the second in the relation to the marker), respectively.

5.4. On the Meaning of the Psychic Factors Φ, $\mathfrak{A}$, σ_1, and σ_2

> ... On the contrary, every time we speak about the cause, there exists something antinotional, undetermined.
>
> Jacques Lacan [9] 28

It was seen how four kinds of speech can be constructed and understood to mark the main psychical (in fact, metaphysically informational or informationally metaphysical) factors Φ, $\mathfrak{A}$, σ_1, and σ_2. According to Lacan, these factors can be in the described cyclic relation and each of them is fixed on the position against the other.

Let us explain the split subject Φ marking the part of information which observes and

comprehends (experiences) itself. In this self-comprehension, Φ experiences its own sense and identity, however, observes also its disaffection to itself (counter-information) within the domain of wish. This constitution of Φ is the consequence of Φ's subordination to categories of symbolic order or language. As a speaking or discursive being, Φ identifies itself in and through the language. On the other hand, Φ feels its own being (informational nature) as unspeakable or informationally connected with that what language to some degree can confirm, but cannot capture it. This informational process is experienced as distress of Φ's being. Thus, subject Φ is split between the marker-master σ_1, which imparts the sense, and remnant $\mathfrak{A}$, which embodies being and cannot be adequately informationally represented (understood).

σ_1 marks the marker-master and represents any marker information to which or against which Φ as information is identified. Subject Φ invests σ_1 in a way where the marker information functions as the last truth: if Φ is confronted with the marker-master σ_1, it does not feel (inform) anymore a need for additional observation, explanation, or excuse (counter-informing). For the subject Φ, the marker-master σ_1 has a sense, which is self-evident; it is a value existing without the need to be spoken about. According to Lacan, these are the concepts of "ego", "unconscious", and "imagination (fantasy)", used by psychoanalysts.

σ_2 marks the knowledge (or belief), which is the discriminating system of language or of linguistic code and which, according to Lacan, is structured by informational iteration of σ_1, i.e., by the conquering power, performed by several markers-masters within the discriminating (synchronous) displacement of all other markers.

The object $\mathfrak{A}$ has some characteristics of the order of the imaginary and of the real. The remnant $\mathfrak{A}$ marks a part of metaphysics (of a being's total information), a part of autopoietically embodied human being, which is not closed under categories of symbolic order and performs non-symbolically (for instance, signal-informationally or molecular-phenomenologically) too. The remnant $\mathfrak{A}$ marks a disorder which obstructs and indirectly confirms the symbolic and imaginary. As a remnant, $\mathfrak{A}$ is the cause of wish.

According to some Lacanian schemes of discourse [1] it is possible to construct various informational relations (operations) existing within each of four types of Lacanian discourse, since the four psychical factors are also in a specific cyclic relation. Thus, besides the basic relation $(\vartheta \,/\, \tau) \models (\rho \,/\, \lambda)$, where $\models$ seems to be a dual (two-way) operator, additionally a general, dynamically structured cyclic scheme of the form

L10. $((((\vartheta \models_1 \rho) \models_2 (\rho \,/\, \lambda)) \models_3 \lambda) \models_4 \tau) \models_5 (\vartheta \,/\, \tau)$

or similar to this form is proposed as a consequence of Lacan's graphic schemes accompanying his philosophy of discourse. In this scheme, $\models_1$ and $\models_2$ can mark a kind of informational incapability or particular non-informing (for instance, within master's and analyst's discourse) and $\models_3$, $\models_4$, and $\models_5$ can mark informational weakness (debility) (for instance, within university and hysteric's discourse). As one can understand, this cycle closes (in an intelligent way) via entities ϑ and $\vartheta \,/\, \tau$. The last formula is the example how basic Lacanian schemes can be formally decomposed according to Lacan's philosophy, getting more and more detailed "algorithms" for informational treatment of the subject.

5.5. The Phantasm as Information

> ... Since the unconscious shows us the abyss through which neurosis is reconciled with the real - with the real which could also be undetermined.
>
> Jacques Lacan [9] 28

As a consequence of discourse a particular informational form appears and informs during the discourse, which can impact and can be impacted by the governing discursive information (informational kernel) within several types of discourse. This specific informational product will be called phantasm. Phantasm as information plays one of the central roles in Lacanian concept of discourse.

Let us proceed from the Lacanian formal expression

L11. $\Phi \diamond \mathfrak{A}$

which marks (an informationally quasi-symmetric) operation or relation between the split subject Φ and its object (remnant) $\mathfrak{A}$. For the mathematically oriented reader it might be not quite clear what do the psychic factors Φ and $\mathfrak{A}$ in fact represent, however, in the course of psychic (or psychoanalytic) investigation these factors can be always informationally decomposed to the needed or conceptually appropriate detail. As Lacan proposes, object $\mathfrak{A}$ is the sliding or level into which that is embedded, what represents the wish of the subject $\mathfrak{S}$.

Further, the meaning of operator $\diamond$ can be determined as 'fantasizes', thus, $\Phi \diamond \mathfrak{A}$ is read as Φ fantasizes $\mathfrak{A}$. According to the general sense of informational operators, it is even possible to introduce a more general formula of phantasm, i.e.,

L12.

$$\Phi_\alpha, \Phi_\beta, \ldots, \Phi_\gamma \diamond \mathfrak{A}_\xi, \mathfrak{A}_\eta, \ldots, \mathfrak{A}_\zeta$$

which can have, for instance, the following meanings: informational entities (split subjects) $\Phi_\alpha, \Phi_\beta, \ldots, \Phi_\gamma$ fantasize, imagine, wish, etc. informational entities (their objects, remnants) $\mathfrak{A}_\xi, \mathfrak{A}_\eta, \ldots, \mathfrak{A}_\zeta$. According to Lacan, it is characteristic that the entities on the left side of operator $\diamond$ are split; it means that these subjects (split informings) perform (inform) as parallel informational entities in themselves.

It is to understand that $\diamond$ is a two-way or quasi-symmetric operation, thus, if the left entity fantasizes the right one, then the right entity also informationally (fantastically) impacts the left one. So, the implication

L13. $\qquad (\$ \diamond \mathfrak{A}) \Rightarrow (\$, \mathfrak{A} \models \$, \mathfrak{A})$

would be appropriate, in general.

It is also to understand that $\mathfrak{A}$ stands against $\$$. This relation is one of the constituents of the psychic economy and is called phantasm. The wish which has to be embedded as information finds its support in phantasm, which is the substrate of the wish, its imaginary regulation. Phantasm appears as a secret, unrevealed informational entity. In fact, phantasm behaves as something informationally ambiguous and paradoxical, for on one side of the phantasmatic operator $\diamond$ there is the last joint of the wish and on the other side something which is informationally embedded into awareness. Thus, phantasm as information belongs to a perverse category, to the domain of absurdity.

Phantasm receives its informational function in the unconscious. If it transits to the level of message, a characteristic situation occurs. Phases, within which phantasm transits, belong to the order of pathologic.

5.6. The University Discourse

> ... The main term, in fact, is not the truth. It is Gewissheit, the certainty.
>
> Jacques Lacan [9] 41

The value of cyclic transformation of matrix L8 for the critics of culture lies in the possibility to understand the manipulation of receptors through messages and the transformation of receptors' metaphysics. It is possible to consider the type of interpellation caused by the main four processes of discourse being identified by Lacan. For instance, the university discourse confronts its receptors with the totalitarian system of knowledge or belief σ_2, by which knowledge is assumed as given. To be able to understand the message, receptors have to be emptied of their own knowledge or belief σ_2, thus producing the state of alienation $\$$. Within the settlement of symbolic order, the receptors do not have any possibility of influence, for σ_1 and σ_2 are under the protection of the transmitter (teacher, ideologist). In principle, this is the place from which one begins to learn speech and to which one returns if it tries to comprehend the totalitarian (predominantly ideologically structured) system. Examples of this situation are students in the system of knowledge or belief and socially subordinated individuals in the system of government or bureaucracy.

Let us examine the obvious two-way two-subject discourse of the form $\alpha, \beta \models \alpha, \beta$ through its mapping onto the scheme of university discourse $(\sigma_2 \,/\, \sigma_1) \models (\mathfrak{A} \,/\, \$)$, by which the basic Lacanian discursive scheme $(\vartheta \,/\, \tau) \models (\rho \,/\, \lambda)$ is appropriated. Thus, let us consider, for example, the four basic informational processes $\alpha \models \alpha$, $\alpha \models \beta$, $\beta \models \alpha$, and $\beta \models \beta$ with their informing, counter-informing and embedding and the simplified scheme L3' with the aim to obtain the feeling how a more detailed (developed or decomposed) scheme would look like.

It is possible to express the adequate cyclic schemes of discourse by means of the so called self-discursive case, for which particular discursive components are determined by B40. Thus, considering the basic positional scheme $(\vartheta \,/\, \tau) \models (\rho \,/\, \lambda)$, there is:

L14.

$$\delta_r(\xi \models \eta) \equiv$$
$$((\vartheta(\xi) \,/\, \tau(\xi)) \models (\rho(\eta) \,/\, \lambda(\eta)); \quad (\rho(\eta) \,/\, \lambda(\eta)) \dashv (\vartheta(\xi) \,/\, \tau(\xi)));$$
$$\delta_r(\xi \models \eta; \omega) \equiv$$
$$((\vartheta(\xi) \,/\, \tau(\xi)) \models (\rho(\mathfrak{C}(\eta)) \,/\, \lambda(\mathfrak{C}(\eta))); \quad (\rho(\mathfrak{C}(\eta)) \,/\, \lambda(\mathfrak{C}(\eta))) \dashv (\vartheta(\xi) \,/\, \tau(\xi));$$
$$(\vartheta(\mathfrak{C}(\xi)) \,/\, \tau(\mathfrak{C}(\xi))) \models (\rho(\omega(\eta)) \,/\, \lambda(\omega(\eta))); \quad (\rho(\omega(\eta)) \,/\, \lambda(\omega(\eta))) \dashv (\vartheta(\mathfrak{C}(\xi)) \,/\, \tau(\mathfrak{C}(\xi))));$$
$$\delta_r(\xi \models \eta; \varepsilon) \equiv$$
$$((\vartheta(\omega(\xi)) \,/\, \tau(\omega(\xi))) \models (\rho(\mathfrak{E}(\eta)) \,/\, \lambda(\mathfrak{E}(\eta))); \quad (\rho(\mathfrak{E}(\eta)) \,/\, \lambda(\mathfrak{E}(\eta))) \dashv (\vartheta(\omega(\xi)) \,/\, \tau(\omega(\xi)));$$
$$(\vartheta(\mathfrak{E}(\xi)) \,/\, \tau(\mathfrak{E}(\xi))) \models (\rho(\varepsilon(\eta)) \,/\, \lambda(\varepsilon(\eta))); \quad (\rho(\varepsilon(\eta)) \,/\, \lambda(\varepsilon(\eta))) \dashv (\vartheta(\mathfrak{E}(\xi)) \,/\, \tau(\mathfrak{E}(\xi))));$$
$$\delta_r(\xi \models \eta) \equiv$$
$$((\vartheta(\varepsilon(\xi)) \,/\, \tau(\varepsilon(\xi))) \models (\rho(\eta) \,/\, \lambda(\eta)); \quad (\rho(\eta) \,/\, \lambda(\eta)) \dashv (\vartheta(\varepsilon(\xi)) \,/\, \tau(\varepsilon(\xi))))$$

for

$$(\xi \models \eta) \in \{\alpha \models \alpha;\ \alpha \models \beta;\ \beta \models \alpha;\ \beta \models \beta\}$$

It is to stress that L14 is a rather simplicistic Lacanian system which shows the arising complexity when additional decomposing (detailing) is performed. In the case of university discourse, entities ϑ, τ, ρ, and λ have to be replaced by entities σ_2, σ_1, $\mathfrak{A}$, and $\$$, respectively. By these replacements in partial discourses of L14, for each partial discourse four subcomponents are obtained, which can be grouped into eight or sixteen processes, respectively, for instance, $\delta_r(\alpha \models \alpha)$; $\delta_r(\alpha \models \beta)$; $\delta_r(\beta \models \alpha)$; $\delta_r(\beta \models \beta)$ and marked by $\delta_u(\alpha, \beta)$, etc. Thus, after the appropriate replacement for the university discourse, there is:

L15.

$$\delta_u(\alpha, \beta) \equiv$$
$$((\sigma_2(\alpha) \,/\, \sigma_1(\alpha)) \models (\mathfrak{A}(\alpha) \,/\, \$(\alpha)); \quad (\mathfrak{A}(\alpha) \,/\, \$(\alpha)) \dashv (\sigma_2(\alpha) \,/\, \sigma_1(\alpha));$$
$$(\sigma_2(\alpha) \,/\, \sigma_1(\alpha)) \models (\mathfrak{A}(\beta) \,/\, \$(\beta)); \quad (\mathfrak{A}(\beta) \,/\, \$(\beta)) \dashv (\sigma_2(\alpha) \,/\, \sigma_1(\alpha));$$
$$(\sigma_2(\beta) \,/\, \sigma_1(\beta)) \models (\mathfrak{A}(\alpha) \,/\, \$(\alpha)); \quad (\mathfrak{A}(\alpha) \,/\, \$(\alpha)) \dashv (\sigma_2(\beta) \,/\, \sigma_1(\beta));$$
$$(\sigma_2(\beta) \,/\, \sigma_1(\beta)) \models (\mathfrak{A}(\beta) \,/\, \$(\beta)); \quad (\mathfrak{A}(\beta) \,/\, \$(\beta)) \dashv (\sigma_2(\beta) \,/\, \sigma_1(\beta)));$$

$\delta_u(\alpha, \beta; \omega) \equiv$
$((\sigma_2(\alpha) / \sigma_1(\alpha)) \models (\mathfrak{A}(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha)));$
$(\mathfrak{A}(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha))) \dashv (\sigma_2(\alpha) / \sigma_1(\alpha));$
$(\sigma_2(\alpha) / \sigma_1(\alpha)) \models (\mathfrak{A}(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta)));$
$(\mathfrak{A}(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta))) \dashv (\sigma_2(\alpha) / \sigma_1(\alpha));$
$(\sigma_2(\beta) / \sigma_1(\beta)) \models (\mathfrak{A}(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha)));$
$(\mathfrak{A}(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha))) \dashv (\sigma_2(\beta) / \sigma_1(\beta));$
$(\sigma_2(\beta) / \sigma_1(\beta)) \models (\mathfrak{A}(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta)));$
$(\mathfrak{A}(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta))) \dashv (\sigma_2(\beta) / \sigma_1(\beta));$
$(\sigma_2(\mathfrak{C}(\alpha)) / \sigma_1(\mathfrak{C}(\alpha))) \models (\mathfrak{A}(\omega(\alpha)) / \Phi(\omega(\alpha)));$
$(\mathfrak{A}(\omega(\alpha)) / \Phi(\omega(\alpha))) \dashv (\sigma_2(\mathfrak{C}(\alpha)) / \sigma_1(\mathfrak{C}(\alpha)));$
$(\sigma_2(\mathfrak{C}(\alpha)) / \sigma_1(\mathfrak{C}(\alpha))) \models (\mathfrak{A}(\omega(\beta)) / \Phi(\omega(\beta)));$
$(\mathfrak{A}(\omega(\beta)) / \Phi(\omega(\beta))) \dashv (\sigma_2(\mathfrak{C}(\alpha)) / \sigma_1(\mathfrak{C}(\alpha)));$
$(\sigma_2(\mathfrak{C}(\beta)) / \sigma_1(\mathfrak{C}(\beta))) \models (\mathfrak{A}(\omega(\alpha)) / \Phi(\omega(\alpha)));$
$(\mathfrak{A}(\omega(\alpha)) / \Phi(\omega(\alpha))) \dashv (\sigma_2(\mathfrak{C}(\beta)) / \sigma_1(\mathfrak{C}(\beta)));$
$(\sigma_2(\mathfrak{C}(\beta)) / \sigma_1(\mathfrak{C}(\beta))) \models (\mathfrak{A}(\omega(\beta)) / \Phi(\omega(\beta)));$
$(\mathfrak{A}(\omega(\beta)) / \Phi(\omega(\beta))) \dashv (\sigma_2(\mathfrak{C}(\beta)) / \sigma_1(\mathfrak{C}(\beta))));$
$\delta_u(\alpha, \beta; \varepsilon) \equiv$
$((\sigma_2(\omega(\alpha)) / \sigma_1(\omega(\alpha))) \models (\mathfrak{A}(\mathfrak{E}(\alpha)) / \Phi(\mathfrak{E}(\alpha)));$
$(\mathfrak{A}(\mathfrak{E}(\alpha)) / \Phi(\mathfrak{E}(\alpha))) \dashv (\sigma_2(\omega(\alpha)) / \sigma_1(\omega(\alpha)));$
$(\sigma_2(\omega(\alpha)) / \sigma_1(\omega(\alpha))) \models (\mathfrak{A}(\mathfrak{E}(\beta)) / \Phi(\mathfrak{E}(\beta)));$
$(\mathfrak{A}(\mathfrak{E}(\beta)) / \Phi(\mathfrak{E}(\beta))) \dashv (\sigma_2(\omega(\alpha)) / \sigma_1(\omega(\alpha)));$
$(\sigma_2(\omega(\beta)) / \sigma_1(\omega(\beta))) \models (\mathfrak{A}(\mathfrak{E}(\alpha)) / \Phi(\mathfrak{E}(\alpha)));$
$(\mathfrak{A}(\mathfrak{E}(\alpha)) / \Phi(\mathfrak{E}(\alpha))) \dashv (\sigma_2(\omega(\beta)) / \sigma_1(\omega(\beta)));$
$(\sigma_2(\omega(\beta)) / \sigma_1(\omega(\beta))) \models (\mathfrak{A}(\mathfrak{E}(\beta)) / \Phi(\mathfrak{E}(\beta)));$
$(\mathfrak{A}(\mathfrak{E}(\beta)) / \Phi(\mathfrak{E}(\beta))) \dashv (\sigma_2(\omega(\beta)) / \sigma_1(\omega(\beta)));$
$(\sigma_2(\mathfrak{E}(\alpha)) / \sigma_1(\mathfrak{E}(\alpha))) \models (\mathfrak{A}(\varepsilon(\alpha)) / \Phi(\varepsilon(\alpha)));$
$(\mathfrak{A}(\varepsilon(\alpha)) / \Phi(\varepsilon(\alpha))) \dashv (\sigma_2(\mathfrak{E}(\alpha)) / \sigma_1(\mathfrak{E}(\alpha)));$
$(\sigma_2(\mathfrak{E}(\alpha)) / \sigma_1(\mathfrak{E}(\alpha))) \models (\mathfrak{A}(\varepsilon(\beta)) / \Phi(\varepsilon(\beta)));$
$(\mathfrak{A}(\varepsilon(\beta)) / \Phi(\varepsilon(\beta))) \dashv (\sigma_2(\mathfrak{E}(\alpha)) / \sigma_1(\mathfrak{E}(\alpha)));$
$(\sigma_2(\mathfrak{E}(\xi)) / \sigma_1(\mathfrak{E}(\xi))) \models (\mathfrak{A}(\varepsilon(\alpha)) / \Phi(\varepsilon(\alpha)));$
$(\mathfrak{A}(\varepsilon(\alpha)) / \Phi(\varepsilon(\alpha))) \dashv (\sigma_2(\mathfrak{E}(\xi)) / \sigma_1(\mathfrak{E}(\xi)));$
$(\sigma_2(\mathfrak{E}(\beta)) / \sigma_1(\mathfrak{E}(\beta))) \models (\mathfrak{A}(\varepsilon(\beta)) / \Phi(\varepsilon(\beta)));$
$(\mathfrak{A}(\varepsilon(\beta)) / \Phi(\varepsilon(\beta))) \dashv (\sigma_2(\mathfrak{E}(\beta)) / \sigma_1(\mathfrak{E}(\beta))));$
$\delta_u(\alpha, \beta) \equiv$
$((\sigma_2(\varepsilon(\alpha)) / \sigma_1(\varepsilon(\alpha))) \models (\mathfrak{A}(\alpha) / \Phi(\alpha));$
$(\mathfrak{A}(\alpha) / \Phi(\alpha)) \dashv (\sigma_2(\varepsilon(\alpha)) / \sigma_1(\varepsilon(\alpha)));$
$(\sigma_2(\varepsilon(\alpha)) / \sigma_1(\varepsilon(\alpha))) \models (\mathfrak{A}(\beta) / \Phi(\beta));$
$(\mathfrak{A}(\beta) / \Phi(\beta)) \dashv (\sigma_2(\varepsilon(\alpha)) / \sigma_1(\varepsilon(\alpha)));$
$(\sigma_2(\varepsilon(\beta)) / \sigma_1(\varepsilon(\beta))) \models (\mathfrak{A}(\alpha) / \Phi(\alpha));$
$(\mathfrak{A}(\alpha) / \Phi(\alpha)) \dashv (\sigma_2(\varepsilon(\beta)) / \sigma_1(\varepsilon(\beta)));$
$(\sigma_2(\varepsilon(\beta)) / \sigma_1(\varepsilon(\beta))) \models (\mathfrak{A}(\beta) / \Phi(\beta));$
$(\mathfrak{A}(\beta) / \Phi(\beta)) \dashv (\sigma_2(\varepsilon(\beta)) / \sigma_1(\varepsilon(\beta))))$

etc. The dynamic scheme of two-way two-participant university discourse $\mathfrak{D}_u(\alpha, \beta)$ can be expressed according to formula B38 by

L16. $((\delta_u(\alpha, \beta) \models \delta_u(\alpha, \beta; \omega)) \models \delta_u(\alpha, \beta; \varepsilon)) \models \delta_u(\alpha, \beta)$

This, rather simplistic case of university discourse shows how complex scenarios of discourse can be constructed. Systems L15 and L16 represent an informational skeleton on which further decompositions can be hanged, coupled, and developed according to imagined purposes. The last case of possible discourse also undoubtedly explicates the importance of joining and combining several concepts - Lacanian and informational. It suggests how it would be possible to structure and organize discursively parallel processes by an informational neural network and programming. And it offers feeling how technological approach of discourse might go behind natural discursive systems and surpass them in complexity as well as possibility.

5.7. The Master's Discourse

The so-called master in Lacanian discourse is a kind of kernel information around which a particular arising of information or informing of kernel information comes into existence.

The master's discourse confronts the receptor through the abiding by distinguished markers-masters σ_1, compelling the receptor that if it should understand the master's message, it has to cease playing of knowledge σ_2, by which all phenomena are presented and explained on the basis of principles embodied by these markers-masters. This effect arises when, for instance, professionals acting in any domain (religious, political, academic, scientific, etc.) read phenomena belonging to their professional domains by notions of some given principled concepts.

The formal discursive components $\delta_m(\alpha, \beta)$, $\delta_m(\alpha, \beta; \omega)$, and $\delta_m(\alpha, \beta; \varepsilon)$, belonging to the master's discourse $\mathfrak{D}_m$, are obtained by circular shifting of the sequence σ_2, σ_1, Φ, $\mathfrak{A}$ into sequence σ_1, Φ, $\mathfrak{A}$, σ_2 in informational system L15, when university discourse transits into master's discourse. Thus, for example,

$\delta_m(\alpha, \beta; \omega) \equiv$
$((\sigma_1(\alpha) / \Phi(\alpha)) \models (\sigma_2(\mathfrak{C}(\alpha)) / \mathfrak{A}(\mathfrak{C}(\alpha)));$
$(\sigma_2(\mathfrak{C}(\alpha)) / \mathfrak{A}(\mathfrak{C}(\alpha))) \dashv (\sigma_1(\alpha) / \Phi(\alpha));$
$(\sigma_1(\alpha) / \Phi(\alpha)) \models (\sigma_2(\mathfrak{C}(\beta)) / \mathfrak{A}(\mathfrak{C}(\beta)));$
$(\sigma_2(\mathfrak{C}(\beta)) / \mathfrak{A}(\mathfrak{C}(\beta))) \dashv (\sigma_1(\alpha) / \Phi(\alpha));$
$(\sigma_1(\beta) / \Phi(\beta)) \models (\sigma_2(\mathfrak{C}(\alpha)) / \mathfrak{A}(\mathfrak{C}(\alpha)));$
$(\sigma_2(\mathfrak{C}(\alpha)) / \mathfrak{A}(\mathfrak{C}(\alpha))) \dashv (\sigma_1(\beta) / \Phi(\beta));$
$(\sigma_1(\beta) / \Phi(\beta)) \models (\sigma_2(\mathfrak{C}(\beta)) / \mathfrak{A}(\mathfrak{C}(\beta)));$
$(\sigma_2(\mathfrak{C}(\beta)) / \mathfrak{A}(\mathfrak{C}(\beta))) \dashv (\sigma_1(\beta) / \Phi(\beta));$
$(\sigma_1(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha))) \models (\sigma_2(\omega(\alpha)) / \mathfrak{A}(\omega(\alpha)));$
$(\sigma_2(\omega(\alpha)) / \mathfrak{A}(\omega(\alpha))) \dashv (\sigma_1(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha)));$
$(\sigma_1(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha))) \models (\sigma_2(\omega(\beta)) / \mathfrak{A}(\omega(\beta)));$
$(\sigma_2(\omega(\beta)) / \mathfrak{A}(\omega(\beta))) \dashv (\sigma_1(\mathfrak{C}(\alpha)) / \Phi(\mathfrak{C}(\alpha)));$
$(\sigma_1(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta))) \models (\sigma_2(\omega(\alpha)) / \mathfrak{A}(\omega(\alpha)));$
$(\sigma_2(\omega(\alpha)) / \mathfrak{A}(\omega(\alpha))) \dashv (\sigma_1(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta)));$
$(\sigma_1(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta))) \models (\sigma_2(\omega(\beta)) / \mathfrak{A}(\omega(\beta)));$
$(\sigma_2(\omega(\beta)) / \mathfrak{A}(\omega(\beta))) \dashv (\sigma_1(\mathfrak{C}(\beta)) / \Phi(\mathfrak{C}(\beta))))$

etc. In this way, the master's discourse $\mathfrak{D}_m$ can be expressed as

L17. $((\delta_m(\alpha, \beta) \models \delta_m(\alpha, \beta; \omega)) \models \delta_m(\alpha, \beta; \varepsilon)) \models \delta_m(\alpha, \beta)$

The complexity of master's discourse is similar

to that of university discourse.

5.8. The Hysteric's Discourse

> ... the subject in position of the eclipse, of vanishing, Φ.
>
> Which is that position? Lately, I determined it by the term fading. I choose this word because of several philological and other reasons, but also because it became familiar at the use of our communication equipment. Namely, fading is that ... when the voice disappears, evaporates and then appears again ... But ... this is only a metaphor for which we have to trace the actual coordinates.
>
> Jacques Lacan [8] 59

The hysteric's discourse of intruding wish Φ demands that the receptor resumes the marker-master σ_1, which gives the meaning and identity to the outward incoherence of the message. But, since the marker-master cannot exist outside of the domain of articulation, this fortifying of markers-masters triggers the constructing of knowledge or belief system σ_2. Examples of such request can be found in trials of consolation, including in endeavors of parents and therapeutists when consoling the exited children and patients.

The formal discursive components $\delta_h(\alpha, \beta)$, $\delta_h(\alpha, \beta; \omega)$, and $\delta_h(\alpha, \beta; \varepsilon)$, belonging to the hysteric's discourse $\mathfrak{D}_h$, are obtained by circular shifting of the sequence σ_2, σ_1, Φ, $\mathfrak{A}$ into sequence Φ, $\mathfrak{A}$, σ_2, σ_1 in informational system L15, when university discourse transits into master's discourse and this one into hysteric's discourse. Thus, for example,

$$
\begin{aligned}
&\delta_h(\alpha, \beta; \varepsilon) \equiv \\
&((\Phi(\omega(\alpha)) \,/\, \mathfrak{A}(\omega(\alpha))) \models (\sigma_1(\mathfrak{E}(\alpha)) \,/\, \sigma_2(\mathfrak{E}(\alpha)))); \\
&(\sigma_1(\mathfrak{E}(\alpha)) \,/\, \sigma_2(\mathfrak{E}(\alpha))) \dashv (\Phi(\omega(\alpha)) \,/\, \mathfrak{A}(\omega(\alpha)))); \\
&(\Phi(\omega(\alpha)) \,/\, \mathfrak{A}(\omega(\alpha))) \models (\sigma_1(\mathfrak{E}(\beta)) \,/\, \sigma_2(\mathfrak{E}(\beta)))); \\
&(\sigma_1(\mathfrak{E}(\beta)) \,/\, \sigma_2(\mathfrak{E}(\beta))) \dashv (\Phi(\omega(\alpha)) \,/\, \mathfrak{A}(\omega(\alpha)))); \\
&(\Phi(\omega(\beta)) \,/\, \mathfrak{A}(\omega(\beta))) \models (\sigma_1(\mathfrak{E}(\alpha)) \,/\, \sigma_2(\mathfrak{E}(\alpha)))); \\
&(\sigma_1(\mathfrak{E}(\alpha)) \,/\, \sigma_2(\mathfrak{E}(\alpha))) \dashv (\Phi(\omega(\beta)) \,/\, \mathfrak{A}(\omega(\beta)))); \\
&(\Phi(\omega(\beta)) \,/\, \mathfrak{A}(\omega(\beta))) \models (\sigma_1(\mathfrak{E}(\beta)) \,/\, \sigma_2(\mathfrak{E}(\beta)))); \\
&(\sigma_1(\mathfrak{E}(\beta)) \,/\, \sigma_2(\mathfrak{E}(\beta))) \dashv (\Phi(\omega(\beta)) \,/\, \mathfrak{A}(\omega(\beta)))); \\
&(\Phi(\mathfrak{E}(\alpha)) \,/\, \mathfrak{A}(\mathfrak{E}(\alpha))) \models (\sigma_1(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha)))); \\
&(\sigma_1(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha))) \dashv (\Phi(\mathfrak{E}(\alpha)) \,/\, \mathfrak{A}(\mathfrak{E}(\alpha)))); \\
&(\Phi(\mathfrak{E}(\alpha)) \,/\, \mathfrak{A}(\mathfrak{E}(\alpha))) \models (\sigma_1(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta)))); \\
&(\sigma_1(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta))) \dashv (\Phi(\mathfrak{E}(\alpha)) \,/\, \mathfrak{A}(\mathfrak{E}(\alpha)))); \\
&(\Phi(\mathfrak{E}(\xi)) \,/\, \mathfrak{A}(\mathfrak{E}(\xi))) \models (\sigma_1(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha)))); \\
&(\sigma_1(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha))) \dashv (\Phi(\mathfrak{E}(\xi)) \,/\, \mathfrak{A}(\mathfrak{E}(\xi)))); \\
&(\Phi(\mathfrak{E}(\beta)) \,/\, \mathfrak{A}(\mathfrak{E}(\beta))) \models (\sigma_1(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta)))); \\
&(\sigma_1(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta))) \dashv (\Phi(\mathfrak{E}(\beta)) \,/\, \mathfrak{A}(\mathfrak{E}(\beta))));
\end{aligned}
$$

etc. In this way, the hysteric's discourse $\mathfrak{D}_h$ can be expressed as

L18. $((\delta_h(\alpha, \beta) \models \delta_h(\alpha, \beta; \omega)) \models \delta_h(\alpha, \beta; \varepsilon)) \models \delta_h(\alpha, \beta)$

The complexity of hysteric's discourse is similar to that of university and master's discourse.

5.9. The Analyst's Discourse

> ... $\mathfrak{A}$ corresponds to that toward which the entire modern development of analysis is oriented when it tries to articulate the object and the relation to the object. Within this research is something righteous, that is to say in the sense that the object relation is that what in principle constructs the mode of the comprehension of the world.
>
> Jacques Lacan [8] 59

The analyst's discourse, which is mastered by the remnant $\mathfrak{A}$, confronts the receptor exactly with information of the being, which is not captured by the marker. This discourse calls the receptor's split nature into the foreground, evoking the sensibility for the excluded element $\mathfrak{A}$. This discourse forces the receptor to produce the new marker-master σ_1, which confirms $\mathfrak{A}$ and suppresses the deficit of the subject's being. Examples of this informational process can be found by students, who, within the answering to Socrates' method, articulate aspects of their experience unnoticed until then.

The formal discursive components $\delta_a(\alpha, \beta)$, $\delta_a(\alpha, \beta; \omega)$, and $\delta_a(\alpha, \beta; \varepsilon)$, belonging to the analyst's discourse $\mathfrak{D}_a$, are obtained by circular shifting of the sequence σ_2, σ_1, Φ, $\mathfrak{A}$ into sequence $\mathfrak{A}$, σ_2, σ_1, Φ in informational system L15, when university discourse transits into master's discourse, this one into hysteric's discourse, and further on into analyst's discourse. Thus, for example,

$$
\begin{aligned}
&\delta_a(\alpha, \beta) \equiv \\
&((\mathfrak{A}(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha))) \models (\Phi(\alpha) \,/\, \sigma_1(\alpha)); \\
&(\Phi(\alpha) \,/\, \sigma_1(\alpha)) \dashv \pi\mathfrak{A}(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha))); \\
&(\mathfrak{A}(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha))) \models (\Phi(\beta) \,/\, \sigma_1(\beta)); \\
&(\Phi(\beta) \,/\, \sigma_1(\beta)) \dashv \pi\mathfrak{A}(\varepsilon(\alpha)) \,/\, \sigma_2(\varepsilon(\alpha))); \\
&(\mathfrak{A}(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta))) \models (\Phi(\alpha) \,/\, \sigma_1(\alpha)); \\
&(\Phi(\alpha) \,/\, \sigma_1(\alpha)) \dashv \pi\mathfrak{A}(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta))); \\
&(\mathfrak{A}(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta))) \models (\Phi(\beta) \,/\, \sigma_1(\beta)); \\
&(\Phi(\beta) \,/\, \sigma_1(\beta)) \dashv \pi\mathfrak{A}(\varepsilon(\beta)) \,/\, \sigma_2(\varepsilon(\beta))))
\end{aligned}
$$

etc. In this way, the analyst's discourse $\mathfrak{D}_a$ can be expressed as

L19. $((\delta_a(\alpha, \beta) \models \delta_a(\alpha, \beta; \omega)) \models \delta_a(\alpha, \beta; \varepsilon)) \models \delta_a(\alpha, \beta)$

The complexity of analyst's discourse is similar to that of university, master's, and hysteric's discourse.

5.10. A Generalization of the Scenario Concerning Lacanian Discourse

> ... Here, the real always returns to the same place - to the place where the subject cogitates, where it is not met by res cogitans.
>
> Jacques Lacan [9] 56

The four types of Lacanian discourse are particular discursive processes which within a real behavior can be rotated from one form to another in the course of speech, public appearance, performance, teaching, ideology, etc. In a discursive process, according to Lacan, it is possible to proceed from university to master discourse, from master to hysteric discourse, from hysteric to analytic discourse, from analytic to university discourse, etc.

The four Lacan's schemata show how a particular discourse can manipulate receptors and transform (impact) the informational processes of their metaphysics. But, because of the rotational property of these schemata, it can also be shown how it is possible to intervene with the intention for opposing or counter-informing effectively the subordinating power of a particular (imposed) discourse. Basically, the way of intervention is similar to the obvious way in psychoanalytic treatment, which includes defiance to the dominating factor of patient's message in the way in which it resumes and explicates the "truth" or subjective basis of this factor. This truth then remains the dominant factor and is requested against its "truth" or its basis. The effect of this process is to produce a new discourse every time when it cycled a quarter turn on the ring of psychic factors clockwise (L9), until the cycling reaches the analytical discourse.

Obviously, the process starts by patient's speaking of university discourse. In this case, the first task of analyst is to discover the marker-master or to expose the identification σ_1. Then, this marker is subjected to systematic, totalizing outward of knowledge or belief $\mathfrak{A}$, which dominates in the patient's speech. This informing transforms the patient's message into master's discourse. Afterwards, the analyst requests the master discourse which, according to Lacan, is identified as a discourse of the self-identical ego, to discover the alienation and split Φ under the monolithic identity σ_1. The result of this process is the next quarter of the cycle which leads to hysteric's discourse. In answering to hysteric's discourse, the analyst does not bid the marker-master σ_1 and knowledge σ_2 requested by the patient as an answer to questions, which concern a distinct identity (for example, sexual identity or death). Instead of this, the analyst turns the patient's question to discover the remnant $\mathfrak{A}$, which is the cause of the patient's wish. Through this process, the analyst discovers the phantasm

$$\Phi \diamond \mathfrak{A}$$

which for the patient functions as an unanalyzable solid belief. From this discourse, the patient as receptor, which answers to the remnant $\mathfrak{A}$, is capable to produce a new marker-master σ_1; this marker is valid as a supposition of new ideal of the ego and as a changed place in the symbolic order, so it can deliver alternatively new opinions (beliefs), different values and even changed identity.

Within a Lacanian discourse, the phantasm of transmitter and receptor can be brought into the formal system as an autonomous, but by other processes impacted informational process. Thus, the following resulting (parallel) informational system can be appropriated:

L20.

$$\mathfrak{D}_\xi \equiv$$
$$(((\delta_\xi(\alpha, \beta) \models \delta_\xi(\alpha, \beta; \omega)) \models \delta_\xi(\alpha, \beta; \varepsilon)) \models \delta_\xi(\alpha, \beta));$$
$$\xi \in \{u, m, h, a\};$$
$$(((\mathfrak{D}_u(\alpha, \beta) \models \mathfrak{D}_m(\alpha, \beta)) \models \mathfrak{D}_h(\alpha, \beta)) \models \mathfrak{D}_a(\alpha, \beta)) \models \mathfrak{D}_u(\alpha, \beta);$$
$$\Phi(\alpha) \diamond \mathfrak{A}(\alpha);$$
$$\Phi(\beta) \diamond \mathfrak{A}(\beta)$$

The first formula in this system describes the dynamic structure of a particular discourse, marked by $\mathfrak{D}_\xi$. The second formula particularizes four types of discourse, $\mathfrak{D}_u$, $\mathfrak{D}_m$, $\mathfrak{D}_h$, and $\mathfrak{D}_a$, known as university, master's, hysteric's, and analyst's discourse, respectively. Finally, the third formula describes the dynamic structure of Lacanian discourse. The first and the third formula are cyclic, so, always the cyclic transition from one to another particular (informational, counter-informational, embedding) and global (university, master's, hysteric's, analyst's) discourse is possible. The fourth and fifth formula mark the presence of transmitter and receptor phantasm during the composed discourse. Certainly, these phantasms change (arise) when discourse transits from one phase to another.

REFERENCES

1. Miller, J.-A., Domination, Education, Analysis, Five Lectures on Lacan in Caracas (in Slovene), Analecta, Univerzum, Ljubljana (1983); also in Cinco conferencias caraquenas sobre Lacan, Editorial Ateneo, Caracas (1980).

2. Šikić, Z., Newer Philosophy of Mathematics (in Serbo-Croatian), Nolit, Beograd (1987).

3. Železnikar, A.P., Informational Logic I, Informatica 12 (3) (1988) 26-38.

4. Železnikar, A.P., Informational Logic II, Informatica 12 (4) (1988) 3-20.

5. Železnikar, A.P., Informational Logic III, Informatica 13 (1) (1989) 25-42.

6. Železnikar, A.P., Informational Logic IV, Informatica 13 (2) (1989) 6-23.

7. Bracher, M., Lacan's "Four Discourses" and the Question of Abortion (in Slovene), Mladina (1989), Nr. 16 (May 5) 34-36.

8. Lacan, J., Hamlet (in Slovene), Analecta, Ljubljana (1988).

9. Lacan, J., The 11-th Seminary, Four Basic Notions of Psychoanalysis (in Croato-Serbian), Naprijed, Zagreb (1986).

10. Železnikar, A.P., Principles of Information, Informatica 11 (1987) 3, 9-17 [by a correction in Informatica 11 (1987) 4, 26] and Cybernetica 31 (1988) 2, 99-122.

REMARKS TO THE CONCLUSION OF THE ARTICLE

In the conclusion of the article (Informational Theory of Discourse II) the following topics will be discussed:

- Pseudo-lacanian forms of discourse

- discourse in the context of modi informationis:

  - discourse as inferring, reasoning, or concluding
  - discourse as modus ponens
  - discourse as modus tollens
  - discourse as modus rectus
  - discourse as modus obliquus
  - discourse as modus procedendi
  - discourse as modus operandi
  - discourse as modus vivendi
  - discourse as modus possibilitatis
  - discourse as modus necessitatis

- scientific discourse:

  - knowledge and truth
  - belief and doctrinaire disciplinarity
  - awareness and consciousness
  - commonsense and scientific reasoning

- master's discourse (or his/hers master's voice):

  - ideological discourse
  - demagogic discourse
  - discourse of the Slavic antithesis
  - political discourse as ideology, demagogy, and antithesis

etc.

Topics of the conclusion will be presented in a rather formal way to show the possibilities of abstract, i.e. (machine-like) informational rigorousness.

INFORMATICA 4/89

ASSURING NUMERICAL STABILITY IN THE PROCESS OF MATRIX REFACTORIZATION WITHIN LINEAR PROGRAMMING PACKAGE ON PC

Keywords: linear programming, HR matrix, matrix factorization, supersparsity, microcomputers

Janez Barle and Janez Grad
Ekonomska fakulteta Borisa Kidriča, Ljubljana

ABSTRACT: One of the most challenging tasks of those who develop linear programming software is development of quick, efficient and reliable matrix refactorization subroutine. The paper describes the implementation of this subroutine within the PC-LIP programming package, which we developed for the IBM-PC personal computers. A major design criterium for PC-LIP was to combine storage economy with numerical stability. The former was achieved using data structures which exploit super-sparsity and the latter implementing state of the art algorithms for basis matrix refactorization. These algorithms were combined with different tools for improving numerical stability. The resulting subroutine performs satisfactory even on a badly scaled data which are quite common in practice.

ZAGOTAVLJANJE NUMERIČNE STABILNOSTI MED REFAKTORIZACIJO BAZNE MATRIKE V PROGRAMSKEM PAKETU ZA LINEARNO PROGRAMIRANJE NA OSEBNEM RAČUNALNIKU: Razvoj hitrega, učinkovitega in zanesljivega podprograma za faktorizacijo bazne matrike spada med najbolj zahtevne naloge pri izgradnji programske opreme za linearno programiranje. Članek podaja opis implementacije tega podprograma v okviru programskega paketa PC-LIP, ki smo ga razvili za IBM kompatibilne osebne računalnike. Pri načrtovanju programskega paketa PC-LIP je bil glavni cilj vskladitev ekonomične izrabe pomnilnika z numerično stabilnostjo. To je bilo doseženo predvsem z uporabo podatkovnih struktur, ki izrabljajo hiperrazpršenost, in najbolj učinkovitih sodobnih algoritmov za izvajanje refaktorizacije bazne matrike. Ti algoritmi so bili kombinirani z različnimi postopki za zagotavljanje numerične stabilnosti. Razviti podprogram za refaktorizacijo je bil uspešen tudi na slabo pogojenih problemih, ki so v praksi dokaj pogosti.

Introduction

A major concern of those who develop linear programming software is how to produce efficient, reliable and numericaly stable computational procedures for solving large-scale problems. When microcomputer software is considered, the problem of fitting algorithms and data structures within the limited storage is also very important. Contemporary literature on computational linear programming offers a plethora of different methods for achieving these goals. Roughly speaking these algorithms and techniques can be divided into following groups:

i) Data structures which are designed for exploiting sparsity in LP data. They can be also tailored for utilization of structure and distribution of nonzero elements contained in LP matrix.

ii) Revised simplex algorithm with product form factorization of basis matrix. State of the art implementations of this method are based on LU factorization.

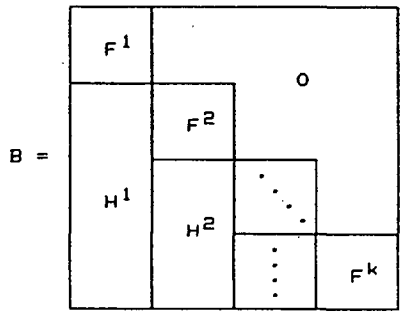
iii) Subroutines for refactorization of basis matrix. They are designed for controlling size and accuracy of product form factorization of basis matrix during the LP solving process.

Each of these groups offers a great choice of

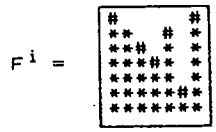
more or less elaborate techniques or algorithms. Sometimes it is not practical to use only the most advanced methods. For example, it is often desirable to sacrifice some computational speed in favour of reliability or storage economy. But in any case refactorization subroutine have to do its job correctly. Refactorization subroutine is also very important part of PC-LIP package which we developed for the IBM-PC personal computers. Practical experiences on real life problems show as that this package is capable to solve even very badly scaled problems. We attribute such a performance mainly to the careful implementation of matrix refactorization subroutine. Our implementation can be described as a successful combination of several state of the art numerical methods with tools for controlling numerical stability. That is why description of our refactorization subroutine may be of interest.

Analysing Structure of Basis Matrix

Refactorization subroutine starts with analytical phase, where the structure of matrix nonzero elements is analysed. In our implementation Hellerman and Rarick algorithm (Hellerman, Rarick, 1971) is used for this purpose. This is quite a famous algorithm which is de facto standard for analytical phase implementations. Results of this algorithms are row and column permutations which transform matrix into form of so called HR matrix. Every HR matrix can be, after suitable rearangement of rows and columns, represented in the form of block lower triangular matrix:



where $F^i (i=1,\ldots,k)$ are square matrices and H^i are rectangular matrices. HR matrices are distinguished from general block lower triangular matrices by structure of matrices F^i. These matrices are always nonsingular. When their dimension exceed 1*1 they are called bumps or external bumps and must have structure similar to one on a next picture, where # is symbol for element which must be different from zero and * symbol for element which can be both zero or nonzero.

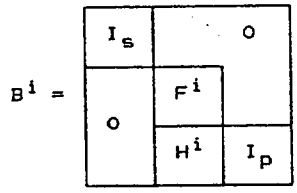


Columns with at least one nonzero above the main diagonal are called spikes. There are two rules concerning spikes:

i) Nonspike columns within bump must have nonzero diagonal elements.
ii) Last column within the bump is a spike having a nonzero uppermost element.

Typical overall number of spikes is much smaller than the number of columns within the matrix. This is an important fact which can be exploited for the economical storing of the factorized basis matrix. It is easy to prove that when product form factorization is formed, only those elementary matrices which correspond to spikes must be actually computed and stored. Other matrices from the product can be replaced by pointers to the non-spike columns (Chvatal, 1983).

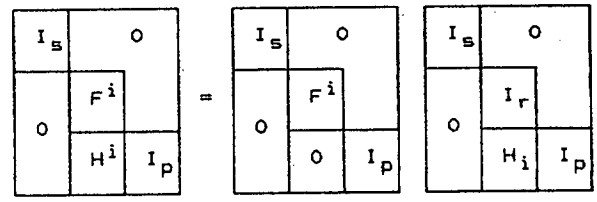
It is easy to check that matrix B with described structure can be represented as a product of matrices having a following form:



where F^i and H^i are situated in the same rows and columns as in matrix B. I_s and I_p are unit matrices of dimension s and p respectively. It is assumed that s and p are numbers of columns to the left and to the right of matrix F^i, which is of dimension r (s+r+p = m). Therefore

$$B = B^1 B^2 \ldots B^k \qquad (1)$$

Adequate definition for this identity can be "generalized product form of matrix B". B^i can be defined as generalized elementary matrices (ordinary elementary matrices, which are contained in product form factorization, can differ from identity matrix only in one column). For matrices structured in such a way the following factorization formula is valid:



(2)

This identity explains why elementary matrices within particular bump can be computed

completely independent from other parts of matrix.

If submatrix F^i is a bump, then factorization (2) is called splitting the bump (Helgason, Kennington, 1982). Main purpose for its usage is to reduce number of nonzero elements in the product form factorization of B. Fill in (creating of new nonzero elements) during the factorization of B^i is restricted to r rows which belong to external bump F^i. It is an improvement if compared with the usual product form factorization, where creation of new nonzero elements is possible in rows belonging to submatrix H^i as well. Experiences show that this approach saves computer storage in spite of some overhead which is necessary to store additional r elementary matrices (Hellerman, Rarick, 1971, Helgason, Kennington, 1982).

Numerical Phase of the Algorithm

Our algorithm for the numerical phase of refactorization which includes splitting the bump is presented in the continuation. This algorithm is modification of recent algorithm (Helgason, Kennington, 1982) in which we included additional techniques for assuring numerical stability. It was necessary due to the fact that in the mentioned algorithm well scaled matrix was assumed. This assumption is in general quite a realistic one because many mainframe programming packages use some procedures for automatic scaling of data prior to applying the revised simplex algorithm. For example, this is true when MPSX/370 is considered (Benichou et al, 1977). However, such kind of procedure is not included in the PC-LIP. We avoided this for the sake of storage economy. The use of scales for rows and columns requires additional storage and, what is more important, practicaly prevents employing of such data structures which take advantage of supersparsity. This is a characteristics of large scale problems which means that the number of distinct numerical values in the problem matrix is usually much smaller than the number of nonzero coefficients (Greenberg, 1976). In the PC-LIP supersparsity is exploited in a standard way: nonzero values within problem matrix are represented by pointers to the table of all distinct nonzero values (Barle, Grad, 1987).

When basis matrix is not well scaled, automatically or by means of proper problem formulation, preassigned pivot can appear to be too small and for this reason inadequate. Two cases, which must be treated differently are:

1. Inadequate pivot is situated within the external bump. In such a case its corresponding column can be treated in a same way as a spike. This means that such columns are included in the process of "spike swapping" (Helgason, Kennington, 1980). In fact this procedure is a variant of partial pivoting which is restricted to spikes within the same bump.
2. Inadequate pivot belongs to column which is outside the bumps (column from the triangle part of the matrix). In this case the only solution is to permute this pivot to the right bottom of the matrix. Such pivots will be referred to as "unstable pivots".

In the continuation of the paper we describe our implementation, which includes handling of above cases.

Algorithm S [Product form factorization for a HR matrix including splitting the bump]

Preassigned sequence of pivots is represented with vector C, consisting of column indices, and vector R, consisting of row indices. It is assumed that these sequences are the results of Hellerman and Rarick's algorithm. Other information obtained with this algorithm can be included into R and C using the following method: indices of spike columns are stored in C with opposite (negative) sign, as well as components of R where external bumps are beginning. Algorithm's input is also basis matrix B, which is of dimension m and parameter TPIVR ("pivot relative tolerance"). All pivots y_r, for which the inequality $|y_r| < TPIVR*y_{max}$ holds, where y_{max} is the largest absolute value of available pivots, are counted as inadequate. Typical values for TPIVR are 0.001 or 0.01.

S0: [Divide the pivots into stable and unstable]
a) Set
   (i) n = m for the number of stable pivots
   (ii) TPIVR = 0.001
   (iii) i = 1
b) If i>m, go to S1.
c) If $R_i<0$, go to S0 g).
d) Set
   (i) r = R_i
   (ii) l = C_i
   (iii) $y_{max} = \max_k |B_{kl}|$
e) If $|B_{rl}| < TPIVR*y_{max}$, set
   (i) for j = i, ..., n-1:
   $R_j = R_{j+1}$ and $C_j = C_{j+1}$
   (ii) R_n = r and C_n = l
   (iii) n = n-1
f) Set i = i+1 and go to S0 b)
g) Set
   (i) t = number of columns in this external bump
   (ii) i = i + t
h) Go to S0 b)

S1: [Initialize]
a) Set
(i) i = 1 (pivot counter)
(ii) l = C_i (pivot column index)
(iii) r = $|R_i|$ (pivot row index)
(iv) j = 1 (counter for elementary matrices)
b) Alocate storage for
(i) y - real vector of dimension m
(ii) ETA-file (data structure for storing matrices E_j).
c) Set y = B_{*l} (l^{th} column of the matrix B).
d) If R_i<0, go to S5.

S2: [Obtain new lower triangular elementary matrix]
a) Set z = y.
Vector z is the r^{th} column of the elementary matrix E_j.
b) Set j = j+1.

S3: [Test for the last stable pivot]
a) If i=n, go to S16.
b) If i<m, set
(i) i = i+1
(ii) l = $|C_i|$
(iii) r = $|R_i|$
(iv) y = B_{*l}

S4: [Test for External Bump]
If R_i>0, go to S2.

S5: [Initialize for Bump]
a) Set
(i) d = j (current length of ETA file)
(ii) p = i (first column in this external bump)
(iii) b = number of columns in this external bump
(iv) t = i+b-1 (last column in this external bump)
b) Set k = p.
c) If k>t, go to S6.
d) If C_k<0, set k = k+1 and go to S5 c).
e) Set
(i) u = C_k
(ii) v = R_k
(iii) $y_{max} = \max_q |B_{qu}|$
f) If $|B_{vu}| < TPIVR*y_{max}$, set $C_k = -C_k$
g) Set k = k+1 and go to S5 c).

S6: [Obtain new lower triangular elementary matrix]
a) Set

$$z_k = \begin{cases} y_k & \text{, for } k = |R_s| \quad (i \le s \le t) \\ 0 & \text{, otherwise} \end{cases}$$

Vector z is the r^{th} column of the elementary matrix E_j.
b) Set j = j+1.

S7: [Test for end of bump]
a) If i=t, go to S11.
b) Set
(i) i = i+1
(ii) l = $|C_i|$
(iii) r = $|R_i|$

S8: [Test for spike]
If C_i>0, set y = B_{*l} and go to S6.

S9: [Spike update]
Solve system of linear equations
$E_d \cdots E_{j-1} y = B_{*l}$

S10: [Swap spikes if $|y_r| < TPIVR*y_{max}$]
a) Compute $y_{max} = \max |y_k|$ for k $\in (R_i, \ldots, R_t)$
b) If $|y_r| \ge TPIVR*y_{max}$, go to S6.
c) Obtain new l, for which l = $|C_s|$ (i<s≤t, C_s<0).
Criteria for choosing l is partial pivoting inside row r.
d) Solve system of equations
$E_d \cdots E_{j-1} y = B_{*l}$.
e) Interchange C_i in C_s and go to S6.

S11: [Obtain new upper triangular elementary matrix]
a) Set

$$z_k = \begin{cases} 1 & \text{, for } k = r \\ y_k & \text{, for } k = |R_s| \quad (s<i) \\ 0 & \text{, otherwise} \end{cases}$$

Vector z is the r^{th} column of the elementary matrix E_j.
b) Set
(i) j = j+1
(ii) i = i-1
(iii) l = $|C_i|$
(iv) r = $|R_i|$

S12: [Test for beginning of bump]
If i=p, set y = B_{*l} and go to S14.

S13: [Test for spike]
a) If C_i>0, set i = i-1 and go to S12.
b) If C_i<0, solve system of equations
$E_d \cdots E_{j-1} y = B_{*l}$
and go to S11.

S14: [Obtain new lower triangular elementary matrix]
a) Set

$$z_k = \begin{cases} 1 & \text{, for } k = r \\ y_k & \text{, for } k = |R_s| \quad (s>t) \\ 0 & \text{, otherwise} \end{cases}$$

Vector z is the r^{th} column of the elementary matrix E_j.
b) Set j = j+1

S15: [Test for end of bump]
a) If i=t, go to S3.
b) If i<t, set
(i) i = i+1
(ii) l = $|C_i|$
(iii) r = $|R_i|$
(iv) y = B_{*l}
c) Go to S14.

S16: [Partial pivoting]
a) If m=n, go to S17.

b) Set i = n+1

c) Sort columns having indices from C_i to C_m in such a way that the number of their nonzero elements is increasing.

d) In the submatrix containing the rows from R_i to R_m and the columns from C_i to C_m perform Gaussian elimination with partial pivoting.

S17: [End]

Product form of B including splitting the bump is obtained.

At the termination of algorithm S, matrix B is represented as a product of elementary matrices E_j:

$$B = E_1 E_2 \cdots E_{j-1} \qquad (3)$$

After obtaining this new factorisation of B, its accuracy must be tested. State of the art method for doing this is the so called Aird-Lynch estimate (Rice, 1985). If this estimate shows that (3) is not accurate enough, algorithm S must be repeated with larger value of TPIVR. In our implementation of algorithm S within the PC-LIP, old value of TPIVR is multiplied by 10.

Partial pivoting within step S10 c) can be performed by using subroutine BTRAN, which can be restricted to only those elementary matrices which belong to the current bump (Saunders, 1976). BTRAN (Backward TRANsformation) is a historical name for the subroutine which solves systems of the form $B^T y = d$, where B is in a product form. The systems of the form $Bz = u$, which appear within several steps of the S algorithm, can be solved by using subroutine FTRAN (Forward TRANsformation). BTRAN and FTRAN are also important subroutines within the revised simplex algorithm.

If row and column permutations determined by R and C are taken into account, all matrices E_i (i=1,2,...,j-1) are either upper triangular or lower triangular, but they are intermixed. That is why (3) is not LU factorization of B. For this reason the revised simplex algorithm with ordinary product form of basis matrix must be applied after performing the refactorization (as it is in PC-LIP). It is not possible to use those algorithms which use and maintain LU format of the basis matrix, for example Forrest and Tomlin method (Forrest, Tomlin, 1972).

If one wishes to use LU format of the basis matrix, splitting the bump can be used only partially on an overall bump or kernel. Kernel is that part of HR matrix which is obtained after the lower and upper triangles have been removed from the matrix. Recently an algorithm was proposed (Helgason, Kennington, 1982) which performs splitting the bump while maintaining the LU format. We briefly sketch how our methods for handling the unstable columns can be incorporated in this algorithm.

By rearangement of rows and columns, the HR matrix may be placed in the following form:

$$B' = \begin{array}{|c|c|c|}\hline U & V & W \\ \hline O & L & T \\ \hline O & M & N \\ \hline \end{array}$$

L and U are lower and upper triangular matrix respectively, O are zero matrices of suitable dimensions. We use T instead of O which is used in the mentioned algorithm (Helgason, Kennington, 1982). This enables transfer of nonstable columns to the rightmost part of the matrix. It is easy to check that for matrix B' the following factorization is valid:

$$B' = \begin{array}{|c|c|c|}\hline I & O & O \\ \hline O & L & T \\ \hline O & M & N \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline U & V & W \\ \hline O & I & O \\ \hline O & O & I \\ \hline \end{array}$$

LU factorization can be performed in usual way for the first matrix at the right hand side. The second matrix is already upper triangular. Due to the fact that a product of two upper triangular matrices is also an upper triangular matrix, LU factorization of B' is obtained.

Conclusions

The matrix refactorization subroutine as described in the paper has been included in the PC-LIP linear programming software package. Our main contribution was that we have combined the already known methods for "splitting the bump" with some methods for assuring numerical stability. The algorithm was tested on many real life problems and proved to be stable even on a very badly scaled data.

Algorithm satisfies also with respect to the computational speed. Unfortunately we have not yet had an opportunity for comparising its performance with some other algorithm performance. It is possible however to measure the amount of reinversion computational time in overall run time. Another interesting test is to examine the effect of inversion frequency on the solution time. We performed these two test on a real life problem with 342 constraints, 454 structural variables and 2048 nonzero elements. With the inversion frequency 20, the optimal solution was obtained after 221 iterations and 565.1 seconds of elapsed time. During the process 12 refactorizations were

performed in 149.7 seconds. This means that refactorizations amounts 26.49% to the overall computational time. We used the same problem for the test with the inversion frequency 50.

The effect of inversion frequency on solution time:

Inversion frequency (iterations)	Solution time (secs)	Iterations	Time per iteration (secs)
20	565.1	221	2.55
50	544	224	2.43

Results show that higher inversion frequency does not effect much the overall solution time. This can be explained with relatively slow execution of the product form variant of revised simplex method. With the use of the Forrest-Tomlin method the performance could be slightly improved (Ashford, Daniel, 1988).

References

1. Ashford R.W., R.C. Daniel: " A note on evaluating LP software for personal computers", European Journal of Operations Research, 35(1988), pp. 160-164.
2. Benichou M., J.M. Gauthier, G. Hentges, G. Ribiere: "The efficient solution of large-scale linear programming problems - some algorithmic techniques and computational results", Mathematical Programming, 13(1977), pp. 280-322.
3. Chvatal V.: Linear Programming, New York - San Francisco, W.H. Freeman and Company 1983.
4. Forrest J.J.H., Tomlin J.A.: "Updated triangular factors of the basis to maintain sparsity in the product form simplex method", Mathematical Programming, 2(1972), pp. 263-278.
5. Greenberg H.J.: "A Tutorial on Matricial Packing", Design and Implementation of Optimization software, Urbino (Italy), (Ed. Greenberg H.J.), Alphen aan den Rijn (Netherlands), Sijthoff and Nordhoff 1978, pp. 109-142.
6. Helgason R.V., Kennington J.L.: "Spike swapping in basis reinversion", Naval Research Logistics Quarterly, 27(1980), pp. 697-701.
7. Helgason R.V., Kennington J.L.: "A note on splitting the bump in an elimination factorization", Naval Research Logistics Quarterly, 29(1982), pp. 169-178.
8. Hellerman E., Rarick D.: "Reinversion with the preassigned pivot procedure", Mathematical Programming, 1(1971), pp. 195-216.
9. Rice J.R.: Numerical Methods, Software, and Analysis, New York, McGraw-Hill 1985.
10. Saunders M.A.: "A fast, stable implementation of the simplex method using Bartels-Golub updating", Sparse Matrix Computations, (Eds. Bunch J.R., Rose D.J.), New York, Academic Press 1976, pp. 213-226.
11. Tomlin J.A.: "On scaling linear programming problems", Mathematical Programming Study, 4(1975), pp. 146-166.
12. Barle J., Grad J.: "PC-LIP: A Microcomputer Linear Programming Package", (program description), Ljubljana, 1987.

FORMAL VERIFICATION OF DISTRIBUTED SYSTEMS

INFORMATICA 4/89

Keywords: distributed system, model, formal specification, verification

Tatjana Kapus, Bogomir Horvat
Tehniška fakulteta Maribor

Distributed systems are inherently concurrent, asynchronous, and nondeterministic. Formal methods and automated tools are needed for helping in describing them without causing a misinterpretation, and in reasoning about their correctness. Different approaches to modelling, formal specification and verification of distributed systems are discussed with respect to their abilities. It is difficult to find a universal formal method. Anyway, a formal approach does not have to be universal for being useful in the design of distributed systems.

Za porazdeljene sisteme je značilno hkratno in asinhrono izvajanje komponent ter nedeterministično obnašanje. Zato potrebujemo formalne metode in računalniško podprta orodja, ki bi nam pomagala popolno in nedvoumno opisati sisteme ter sklepati o njihovi pravilnosti. V članku govorimo o različnih pristopih k modeliranju, formalni specifikaciji in verifikaciji porazdeljenih sistemov glede na njihove zmožnosti. Težko je najti univerzalen formalni pristop. Seveda pa je lahko pristop koristen, čeprav ni vsestranski.

0. Introduction

Informal software design techniques often rely on trial and error involving possibly several implementation and redesign loops. Formal methods and computer-aided tools are needed in the entire design process to avoid this potentially expensive procedure. This is especially true for distributed systems, because they are inherently concurrent, asynchronous, and nondeterministic. Conceptually, they are thought of as being composed of processes which interact by exchanging messages. If the number of possible interactions is large, their behaviour is extremely difficult to reason informally about, and even to describe without causing a misinterpretation.

A variety of formal specification and verification approaches are being investigated. They are based on some model of computation. In this paper we discuss different approaches to modelling, formal specification and verification of distributed systems by asking if they have some desirable abilities. We ask, for example, if they manage state explosion, if it is possible to specify and verify safety and liveness properties, if control and data related properties can be verified, if a proof system based on a model is compositional, which kind of communication can be dealt with, if an approach can be used for different applications, and also, if verification can be easily automated.

Note that throughout the paper we talk about the classical system design approach. There, a requirement specification is stated first, which describes the behaviour of a system from its user's view, without talking about its internal structure. It serves as a contract between the user and the designer. A design specification is obtained by decomposing the system into communicating processes. It is the designer's work to verify if it meets the requirement specification. It is good if every process can be specified separately. The possibility of modular specification and verification of a design against a requirement specification in absence of the code reduces the design complexity of distributed systems.

1. State machines

State machines are widely used to model distributed systems. Component processes are represented by states and possible transitions between them. The transitions represent events - transmissions and receptions of messages. The system's state space can be obtained, such that its states are determined by a state of every component, and its transitions are the

components' ones. Verification can be generally viewed as requiring reasoning about the complete state space of a system /11/. But, as the number of states increases, it becomes a difficult task. We can say that the basic role of formal techniques is in helping the designer to manage the state explosion.

The problem is being solved in two ways. The first one remains in the state-machine concept. After the component processes are formally specified, the system's state space is constructed and examined. Of course, computers are exploited to do it. This is so-called exhaustive analysis. The state space is in fact a reachability graph, and the analysis is also called reachability analysis. The second way is to use mathematical theories built on appropriate models which would not force us into construction of the state space.

One of the problems with exhaustive analysis is that a representation of the complete state space of a system must be constructed. Usually, some transformations have to be performed to obtain a graph of a reasonable size, such as projections, reductions, and selections /11/. They should preserve the behaviour being analyzed. In most cases, transformations are focused on preserving control aspects, and ignoring data aspects of the system. Unfortunately, only some general properties, such as absence of deadlock or livelock, can be proved in this way, or ordering of communication events can be verified. That is why state machines are typically used for verification of communication protocols. Even when projections are used, construction of a reachability graph is time-consuming for large systems. Besides, finite graphs cannot be built sometimes. One solution to the problem, and to make it possible to analyze more system-specific properties, is the use of simulation as a complementary approach. It is in essence exploration of a selected portion of the state space. However, it cannot "prove" properties about the complete state space, but it can increase confidence in the correctness of the system. A much exploited advantage of simulation is that statistical information about the performance of the system may be calculated from the results of a simulated execution. It is said that "exhaustive analysis and simulation are both sides of the same coin" /2/.

Most currently existent computer-aided tools which can be used for design of real scale distributed systems use exhaustive analysis and simulation. One would say that exhaustive analysis and simulation are used because of the lack of appropriate theories. It is true that for many of them only a conceptual framework is provided, mainly concerning communication and concurrency issues, and their use is only shown for small scale problems. Besides, exhaustive analysis and simulation are certainly more easily automated. In the case of finite number of states, algorithmic verification is possible. But there are other reasons. For instance, it is thought that simulation is nearer to the culture of an average computer scientist /6/. And formal proofs can play its role best when a design is clear enough, while the designer needs tools for assisting him in the design process to achieve this stage by letting him precisely express his ideas, and in the first place validate them rapidly by simulation.

Examples of the tools are OVAL /2/, Véda /6/, RGA /11/, SARA /4/. The tools typically use standardized state-machine languages, such as CCITT's Specification and Description Language (SDL) /2/, and ISO's Estelle /6/, many of them use Petri nets /11/ and related formalisms /4/, for description of systems, because a primary concern here is to provide the designer with a precise and expressive formal language which is easy to learn and to use. This is not the case with abstract mathematical formalisms.

Some analysis procedures, such as searching for deadlocked states, can be built into a tool. A question arises, how to express specific requirements, to traverse the reachability graph interactively, and to verify if they are met. One way is to write them in the same formalism as the design being verified /14/. The RGA tool /11/ allows the user to specify first-order logic propositions and predicates about places and transitions of its Petri-net designs, and even to write an algorithm to perform more complex analysis of the design. Temporal logic specifications may be written in some tools.

There is a similar problem with simulation. In Véda /6/, an observer can be defined to observe execution traces of a simulated system instead of the user, and to report errors when requirement specifications are not met. Another question arises concerning simulation. The development of simulation requires "test scenarios" of the system environment. They can be generated in a fully random or in an interactive way. The authors of SARA /4/, for example, have decided to model the environment explicitly, like the system being designed. A well defined behavioural model of the environment then serves to stimulate the system, and to validate its behaviour. We see that simulation can be in general fully automated.

2. Axiomatic approach

When talking about state-machine notations, we should also mention Milner's CCS (Calculus of Communicating Systems) /9/, and Hoare's CSP (Communicating Sequential Processes) /5/, although they do not model states explicitly. They rather describe processes in terms of observable events. Their advantage is that they provide a range of algebraic laws for comparing, and reasoning about distributed systems, so that formal specification and verification can be carried out in the same framework. If component processes of a system are described in CCS or CSP, we can still construct the system's "state space". This is indeed convenient for "finite-state" cyclic systems, because the observable behaviour of

the system can be obtained and simply compared with the system specification for its correctness. But, to avoid a possibly threatening state explosion, other kinds of reasoning have to be employed with state machines, induction on state transitions, for example. CSP offers besides compositional proof rules.

Till now, we have been talking about constructive descriptions of processes. In the constructive approach, also called operational approach, a process is specified as an abstract machine describing a computation. Such a specification is implementation oriented. We specify a program (i.e. a process) essentially by writing another, presumably simpler, program /7/. For requirement specification and formal verification purposes, specifying processes by stating their properties, constraints that any implementation must satisfy, seems more convenient. This is so-called axiomatic approach. In general, there are two kinds of properties. Safety properties express what may happen, or that something bad must not happen. Examples of them are partial correctness, mutual exclusion, and deadlock-freedom. Liveness properties express what must eventually happen, or that a particular good thing must eventually happen. They are temporal properties. Termination and starvation-freedom are liveness properties.

We can talk about the properties in the constructive approach, too. How could we specify a process, if not by describing what may, or what must happen?! Liveness properties cannot be specified in every model. Only safety properties can be stated and verified within the classical state-machine model. If it is not possible to specify what must happen, the second best way is to express what may happen. The difference between the constructive and the axiomatic approach is that in the former we specify a process executions step by step, and in the latter properties are written in form of logic assertions which hold for execution sequences we would get if the process unfolded over time following the constructive description. Unfortunately, algorithmic verification in the axiomatic approach is not possible in general. Proofs have to be designed by hand (if a theorem prover is not available) and a certain ingenuity often is required to find the proof /13/.

Compositionality of proof rules means that proof of the correctness of a compound process can be constructed from proofs of correctnes of its parts. Hence, the system's state space does not need to be constructed. With another word, to prove a property of a program, we do not have to know the complete program, but only requirement specifications of its parts.

Hoare has achieved compositionality by introducing a trace model /5/. A trace of the behaviour of a process is a finite sequence of symbols recording the events in which the process has engaged up to some moment in time. A process in the model satisfies a specification if the specification expression is true for all its possible traces. "Concatenation of sequences", "prefix of a sequence", and "the lenght of a sequence" are basic notations to the trace specifications. A similar approach is used in the compositional proof system for networks of processes of Misra and Chandy /10/. It has to be stressed that safety properties hold for complete execution sequences and their finite prefixes. Some liveness properties are not fulfilled by prefixes, but always hold for complete sequences. Because execution sequences may be infinite, liveness properties are difficult to specify in this model due to finiteness of traces.

Temporal logics are most often used for specifying liveness properties. The temporal operator 'eventually' is especially suitable for expressing progress properties, i.e. that an event will eventually happen. Temporal logic is also used in combination with state-machine model in so-called model checking /13/ for verification of liveness properties. Unfortunately, we need here a finite system state space which we check against temporal logic formulae. Finding models that would allow modular verification of temporal properties, and not only of safety properties, i.e. a compositional proof system for both of them, is a tough problem.

An example of such a model is one that has been found by Nguyen et al. /12/. One would expect that infinite sequences will be used in place of finite traces to model process executions. Instead, a behaviour has been introduced for better modelling of progress and termination or deadlock. It is an infinite sequence of observations. Every observation includes a trace of events that have happened up to the moment of the observation. Like in Hoare's traces, the trace may be at most one event longer in the next moment, i.e. in the next observation. It means that events are totally ordered, and that concurrency is modelled with interleaving of concurrent events. The compositional proof system based on the model uses linear temporal logic /8/. With the introduction of infinite behaviours, i.e. infinite sequences of traces, it has been achieved that the previously mentioned trace notations are still the basic ones, but temporal properties can be stated in terms of them by temporal operators. The model is also interesting because systems with synchronous and asynchronous communication can be specified and verified, which is not usual in other existent compositional proof systems. Communication is synchronous if a process cannot send anything until the receiving process is ready to accept it as input, and it is asynchronous if a process can send an output as soon as it is ready. To enable modular specification and verification of temporal properties for both kinds of communication, it has been necessary to represent the readiness of processes to communicate in the model.

It is not necessary to use temporal logic to express temporal properties. It can be always

replaced by first-order logic with certain relations introduced. The reason it is often used is because it is concise and elegant.

Trace specifications are very suitable for data-flow computations, for example, but seem awkward in expressing properties whose data structures are not well-defined sequences, such as properties in unreliable systems.

Chen and Yeh /3/ have proposed EBS (Event Based Specification Language) which takes the concept of events more fundamental than that of traces, so that unreliable systems can be more easily specified. Partial ordering on events is used, so that not all possible interleavings of potentially concurrent events have to be considered as with total ordering. And it does not use temporal operators. To say that an event will eventually cause the occurrence of another event, it uses a binary relation. Safety and liveness properties can be specified and verified separately like in Nguyen's system, so that verification is less complex.

3. Conclusion

Some approaches to modelling, formal specification and verification of distributed systems have been discussed. We have shown their main characteristics and problems that have to be overcome in searching for new methods. It seems hardly possible to find a universal formal approach. A much exploited solution is in building automated tools which integrate several useful approaches, and do not force their users into procedures unnatural to them. And perhaps it is true that theories for specific applications should be established before going into generalization /1/.

References

/1/ Boute, R.T. (1988), "On the shortcomings of the axiomatic approach as presently used in computer science", CompEuro 88, System design: Concepts, methods and tools, Brussels 1988, Washington 1988, pp. 184-193.

/2/ Cavalli, A.R. and Paul, E. (1988), "Exhaustive analysis and simulation for distributed systems, both sides of the same coin", Distributed Computing, vol. 2, no. 4, pp. 213-225.

/3/ Chen, B.-S. and Yeh, R.T. (1983), "Formal Specification and Verification of Distributed Systems", IEEE Trans. Software Eng., vol. SE-9, no. 6, pp. 710-722.

/4/ Estrin, G., Fenchel, R.S., Razouk, R.R., and Vernon, M.K. (1986), "SARA (System ARchitects Apprentice): Modelling, Analysis, and Simulation Support for Design of Concurrent Systems", IEEE Trans. Software Eng., vol. SE-12, no. 2, pp. 293-311.

/5/ Hoare, C.A.R. (1985), Communicating Sequential Processes, Prentice-Hall International, London.

/6/ Jard, C., Monin, J.-F., and Groz, R. (1988), "Development of Véda, a Prototyping Tool for Distributed Algorithms", IEEE Trans. Software Eng., vol. 14, no. 3, pp. 339-352.

/7/ Lamport, L. (1983), "What good is temporal logic?", Proc. IFIP 83, ed. R.E.A. Mason, North-Holland, pp. 657-668.

/8/ Manna, Z., Pnueli, A. (1981), Verification of concurrent programs, Part 1: The temporal framework, Tech. Rep. STAN-CS-81-836, Stanford University, June 1981.

/9/ Milner, R. (1980), A Calculus of Communicating Systems, Springer - Verlag, Berlin.

/10/ Misra, J. and Chandy, K.M. (1981), "Proofs of Networks of Processes", IEEE Trans. Software Eng., vol. SE-7, no. 4, pp. 417-426.

/11/ Morgan, E.T. and Razouk, R.R. (1987), "Interactive State-Space Analysis of Concurrent Systems", IEEE Trans. Software Eng., vol. SE-13, no. 10, pp. 1080-1091.

/12/ Nguyen, V., Demèrs, A., Gries, D., Owicki, S. (1986), "A model and temporal proof system for networks of processes", Distributed Computing, vol. 1, no. 1, pp. 7-25.

/13/ Pehrson, B. (1989), "Formal Specification Methods", CompEuro 89, Tutorial Sessions, ed. W. Anacker and R. Beyer, Hamburg 1989.

/14/ Rea, K, and Johnston, R.de B. (1987), "Automated Analysis of Discrete Communication Behaviour", IEEE Trans. Software Eng., vol. SE-13, no. 10, pp. 1115-1126.

This work was supported by Research Communities of Slovenia.

CHARACTERIZATION OF CIRCUITS IN GRID OBTAINED BY REGULAR AND SEMI-REGULAR TESSELLATIONS

INFORMATICA 4/89

Joviša Žunić, Novi Sad
Faculty of Engineering
Dr Ivan Stojmenović, Institute of Mathematics, Novi Sad

Keywords: algorithm, circuit, tessellation

ABSTRACT: In this paper circuits in grids which are obtained by using plane tessellations are observed. Isomorphism and congruence of circuits in these grids is defined in natural way. Conection between these relations is discussed.

1. INTRODUCTION

A tesselation of plane is a covering of the plane by using polygons. It is known that there are exactly eleven ways to cover plane by using regular polygons. Three of these are regular tessellations, where each vertex is surrounded by identical regular polygons (see fig.1). The other eight are semi-regular tessellations, in which each vertex is surrounded by an identical cycle of regular polygons (see fig.2).

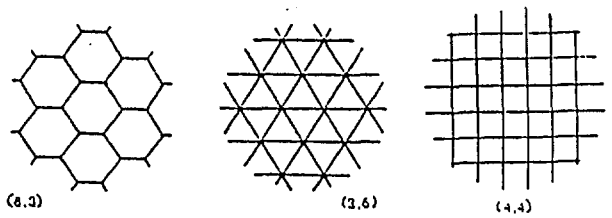


Figure 1. The three regular tesselations

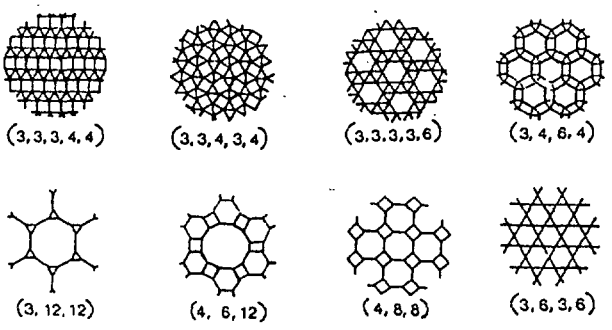


Figure 2. The eight semi-regular tesselations

In this way eleven infinite periodic grids are obtained. (This grids are plane representation of infinite plane graphs.). Let G be one of obtained grids. A circuit of the lenght m in a grid G is a oriented closed path without repeated vertices, containg m edges.

A circuit C in the grid G determines a simple polygon which consists of the edges of C. We will say that a circuit C_1 is congruent to a circuit C_2 iff polygons determined by circuits C_1 and C_2 are congruent polygons. Also, in natural way we define an isomorphism of circuits in the grid G which is obtained by regular and semi-regular tessellations. Let C_1 and C_2 be the circuits in the grid G. Then: the circuits C_1 and C_2 are isomorphic circuits iff there exists a congruence transformation T such that:

1) T maps the grid G into itself and
2) T maps a polygon determined by the circuit C_1 into polygon determined by circuit C_2.

Also we say that simple polygons A and B in the grid G are isomorphic polygons iff circuits determined by A and B are isomorphic circuits.

2. WORD REPRESENTATION OF CIRCUITS

Let G be one of grids obtained by using tessellations. Grid G is periodic. Let us determine period of grid G. If n is the number of edges in period of G then the number of oriented edges is 2n and we shall denote these oriented edges (vectors) by $v(0),v(1),\ldots,v(2n-1)$. In this way for any oriented edge of the grid G there is corresponding uniquely determined vector from the set $v=\{v(0), v(1),\ldots,v(2n-1)\}$.

Let A and B be points in the grid G, and P oriented path of lenght t, from A to B. If path P consists of oriented edges $v(i_1),v(i_2),\ldots,v(i_t)$ respectively, then the word $f(P)=i_1i_2\ldots i_t$ which corresponds to path P is uniquely determined. Specialy, for i=1 $f(v(i))=i$. Let A^k be the set of all words of lenght k over the alphabet $A=\{0,1,\ldots,2n-1\}$ and $A^*=\cup A^k$ $k>0$.

Then denote by A" the set of all words which corresponds to oriented pats in the grid G. That means: $a\in A^*$ <=> exists path P such that f(P)=a.

If the word $a=i_1i_2\ldots i_t$ is from A", then a determines the path $v(i_1)\ldots v(i_t)$ such that f(P)=a. The circuit C of lenght n determines 2n closed oriented paths, depending on the choice of the initial vertex and the orientation of the circuit. A function f maps these 2n oriented paths into 2n words of the set A". Let us denote the set of these 2n words by Q(C) (for circuit C). Let T be isometry which maps grid G into itself. Let $T(v(i))=v(i')$ $i=0,1,2,\ldots,2n-1$ then $(0',1',\ldots,(2n-1)')$ is permutation of $(0,1,\ldots,2n-1)$. Transformation T maps path $P=v(i_1)v(i_2)\ldots v(i_t)$ into path T(P) such that $T(P)=T(v(i_1)v(i_2)\ldots v(i_t))=T(v(i_1))T(v(i_2))\ldots T(v(i_t))=v(i_1')v(i_2')\ldots v(i_t')$ or $f(T(P))=i_1'i_2'\ldots i_t'$.

Let A"' be the set of all words which correspond to circuits in the grid G. Also every word a from A"' $(a=i_1\ldots i_t)$ determines circuit $C=v(i_1)v(i_2)\ldots v(i_t)$ (which determines simple polygon with edges of C).

Let a and b be words from A"' . We say that a and b are in the relation α iff circuits, which are determined by words a and b, are isomorphic circuits.

LEMMA 1: Relation α is equivalence relation.

PROOF: The set τ of all congruence transformations which map grid G into itself is a group.
Specialy: If T=I (identical mapping) then for $a,b\in Q(C)\Rightarrow a\alpha b$.

In the set of vectors $\{v(0),v(1),\ldots,v(2n-1)\}$ we define relation ρ by: $v(i)\ v(j) \Leftrightarrow$ exists isometry T which maps the grid G into itself such that

$$T(v(i)) = T(v(j))$$

LEMMA 2: Relation ρ is a relation of equivalence.

PROOF: Directly from definition.

Also, we say that: $i\rho j$ iff $v(i)\rho v(j)$.

If P is a path from point A to point B then vector $\vec{AB}$ is equal to the vector sum of oriented edges which the path P contains.

LEMMA 3: Word $a=i_1i_2...i_t$ from A" is from A"' (or $v(i_1)v(i_2)...v(i_t)$ is circuit) iff 1) $v(i_1)+v(i_2)+...+v(i_t)=0$ (vector summ) and 2) $v(i_j)+v(i_{j+1})+...+v(i_{j+k})\neq 0$ for $k<j+k\leq t$.

3. ALGORITHM FOR COUNTING NONISOMORPHIC CIRCUITS

Using observations from previous section we can propose one common algorithm for determination numbers of nonisomorphic circuits on each of grids obtained by tessellations (regular and semi-regular).

Let e_1 and e_2 be the vectors from $\{v(0),v(1),...,v(2n-1)\}$ which are not colinear. Now, we determine coordinates of each vector from $\{v(0),v(1),...,v(2n-1)\}$ with respect to vectors e_1 and e_2.

Let $v(i)=\alpha_1e_1+\beta_1e_2$ $(i=0,...,2n-1)$ then follows (from Lemma 3):

LEMMA 4: If $a=i_1i_2,...,i_t$ is word from A"' then

$$\sum_{i=0}^{2n-1}\alpha_i l(i)=0 \quad \text{and} \quad \sum_{i=0}^{2n-1}\beta_i l(i)=0$$

where $l(i)$ is the number of occurences of character i in the word $a=i_1i_2...i_t$. CONDITION 3 will be called CONDITION-G.

LEMMA 5: If P is an oriented path in the grid G then: P is a circuit iff:

1) the word f(P) satisfies CONDITION-G and
2) no subword b of a satisfies CONDITION-G.

We use following input data and their notations:

1) Number of vectors in period of G.

2) Number of continuations of each vector denoted by C. (Vector v(i) is a continuation of a vector v(j) if word ji is from A").

3) Continuations of each vector. Corresponding characters for continuations of vector v(i) denoted by $c(i,1),c(i,2),...,c(i,C)$.

4) Number of initial vectors-denoted by I.Note: Initial vector can be any vector. It is clear that if the word $a=i_1i_2...i_t$. Is from A" and $i_1\rho j_1$ then there exists a word $b=j_1j_2...j_t$ such that $a\alpha b$. That means: the number of initial vectors can be equals to the number of equivalence classes with respect to relation ρ. In this way can be reduce the computation time.

5) Initial vectors-corresponding characters denoted by $Iv(1),Iv(2),...,Iv(I)$.

6) Number of transformations (of τ).

7) One isometry of first class which maps grid into itself.

8) One isometry of second class which maps grid into itself.

9) For every vector v(i) its opposite vector v(j). $(v(i)=-v(j))$.

Note: for grid $3^4.6$ isometry of first class no exists so 7 is identical mapping.

Let $a=i_1i_2...i_t$ be word from A", and let a(j) denoted the word $i_ji_{j+1}...i_t$ $1\leq j\leq t$. If CONDITION-G is not satisfied for any $j(1\leq j\leq t)$ then we call the word $i_1i_2...i_t$ addable. It is clear that $i_1i_2...i_t$ can be completed to a word $i_1i_2...i_m$ $(m>t)$, representing a circuit iff $i_1i_2...i_t$ is addable. Also it is obvious that $a=i_1i_2...i_t$ denotes a circuit iff a(j) satisfied CONDITION-G only for j=1.

All words that are α-equivalent to a word a representing a circuit we can obtain using 6,7,8,9 (see input data). We consider only the equivalent words begining by one of initial vectors (input data-5) and sort them in lexicographic order. We choose the first word a' as a representative of this class. Hence, if the word a is equal to a', then word a represents a circuits of lenght t and print it.

Our algorithm can be conveniently explained using two phases: extend and reduce. These phases correspond to the addable and nonaddable cases respectively.

```
READ (t)
FOR k=1 to I DO
 BEGIN
  i1=Iv(k); m:=1
  REPEAT
  IF i1i2...im is addable
   THEN extend
   ELSE IF i1i2...im is representative of
           a nonisomorphic circuits
           THEN print i1i2...im
   reduce
  UNTIL m=1
 END
where
extend ≡ BEGIN m:=m+1; im:=c(im-1,1) END
reduce ≡ WHILE im=c(im-1C) and m≥2
           DO m:=m-1
        IF m≠1
           THEN BEGIN
                  t:=0
                  REPEAT t:=t+1
                  UNTIL im=c(im-1,t)
                  im:=c(im-1,t+1)
                END
```

Data obtained by proposed algorithm will be given in next section, k(t) denoted the number of nonisomorphic circuits lenght of t,

Grids which is obtained by tessellations $6^3,4^4,3^6$ are not specialy treated, but they have been observed in the papers: |1|,|2|,|3|. The algorithm presented here could be also directly applied to these a rids.

4. CONNECTION BETWEEN ISOMORPHISM AND CONGRUENCE OF CIRCUITS

It is clear that if C_1 and C_2 are isomorphic circuits then C_1 and C_2 are congruent circuits. In this section we will show that for grids obtained by tessellations $3^3.4^2$, $3^2.4.3.4$, $3.4.6.4$, $3.6.3.6$, 3.12^2, $4.6.12$, 4.8^2 (all semi-regular except $3^4.6$) is satisfied: if circuits C_1 and C_2 are congruent circuits then they are isomorphic circuits. For grids obtained by regular tessellations previous statament follows obviously because of that they are not specialy treated.

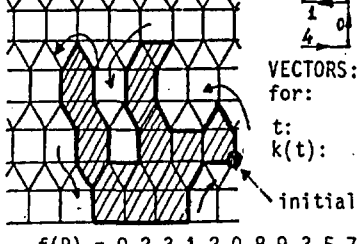
In proofs of following lemmas we will use:

LEMMA 6: Let $M=M_1M_2...M_t$ and $N=N_1N_2...N_t$ be congruent polygons such that $M_{i_1}\equiv N_{j_1}, M_{i_2}\equiv N_{j_2}, M_{i_2}\equiv N_{j_3}$ (for some integres i_1,i_2,i_3 from $(1,2,...,t)$, then: if points M_{i_1},M_{i_2} and M_{i_3} are not colinear then $M_i\equiv N_i$ for all $i\in\{1,2,...,t\}$.

We shall denote by $\measuredangle(i,j)$ the angle between vectors v(i) and v(j). By r(M) will denoted the word $m_1,_2...$ determined by a simpl polygon $M=M_1M_2...M_t$ such that $m_i=$ $f(M_i\vec{M}_{i+1})$.

LEMMA 7: Let G be the grid obtained by tessellation $3^3.4^2$, then: if M and N are congruent polygons in grid G then $r(M)\alpha r(N)$.

PROOF: Equivalence classes with respect to relation ρ are: I={0,5} II={1,4,9,6} III={2,3,7,8} (see fig.3)
Let $M=M_1M_2...M_t$ and $N=N_1N_2...N_t$ are congruent polygons in the grid G and $r(M)=m_1m_2...m_t$ and $r(N)=n_1n_2...n_t$.



VECTORS: 5 6 7 8 9 are opposite
for: 0 1 2 3 4

t:	3	4	5	6	7	8	9	10	11
k(t):	1	2	2	4	6	17	32	90	204

initial vertex

f(P) = 0 2 3 1 2 0 8 9 3 5 7 5 9 0 2 0 2 3 5
7 5 7 5 4 4 4 0 8 4

Fig. 3.

1-case: $m_1,n_1\in I$ then $m_1m_2...m_t\alpha 0m_2'...m_t'$ and $n_1n_2...n_t\alpha 0n_2'...n_t'$ (words $0m_2'...m_t'$ and $0n_2'...n_t'$ exist because m_1 and n_1 are from same equivalence class) if $m_2'=n_2'$ then by Lemma 6 $m_i'=n_i'$ for $i=3,...,t$ if $m_2'\neq n_2'$ then we apply reflection in

a line determined by vector v(0) which maps grid G into itself. The image of $0m_2'...m_t'$ is $0m_2''...m_t''$.

$$\sigma_0 \begin{pmatrix} 0&1&2&3&4&5&6&7&8&9 \\ 0&6&8&7&9&5&1&3&2&4 \end{pmatrix}$$

since $\sphericalangle(0,m_2'')=\sphericalangle(0,n_2')$ we have $m_1''=n_1'$, $m_1m_2...m_t\alpha 0n_2'...m_t\alpha 0n_2''m_3''...m_t''$, $n_1n_2...n_t\alpha 0n_2'...n_t'$ that means (by Lemma 6) $n_3'=m_3'',...,n_t'=m_t''$ or $m_1m_2...m_t\alpha n_1n_2...n_t$.

2-case: $m_1n_1G||$ then $m_1m_2...m_t\alpha 1m_2'...m_t'$ and $n_1n_2...n_t\alpha 1n_2'...n_t'$
$\sphericalangle(1,m_2')=\sphericalangle(1,n_2')\Rightarrow m_2'=n_2'$
If $m_2'=n_2'=1$ then we continue until $m_i'\neq 1$ but then using Lemma 6 $m_i'=n_i'$ or $m_1m_2...m_t\alpha 1m_2'...m_t'=1n_2'...n_t'\alpha n_1n_2...n_t$.

3-case: m_1,n_1GIII then $m_1m_2...m_t\alpha 2m_2'...m_t'$ and $n_1n_2...n_t\alpha 2n_2'...n_t'$
$\sphericalangle(2,m_2')=\sphericalangle(2,n_2')$
if $m_2'=n_2'$ then clearly $m_i'=n_i'$ for $i=3,...,t$ and $m_1m_2...m_t\alpha n_1n_2...n_t$ if $m_2'\neq n_2'$ then, let be for example, $m_2'=3$ and $n_2'=4$, now $\sphericalangle(3,m_3')=\sphericalangle(4,n_3')\Rightarrow m_3'=6$ and $n_3'=3$
$\sphericalangle(2,m_2')=\sphericalangle(2,n_3')$
that means $2m_2'...m_t'=236$ and $2n_2'...n_t'=243$ but $2\,3\,6\,\alpha\,2\,4\,3$

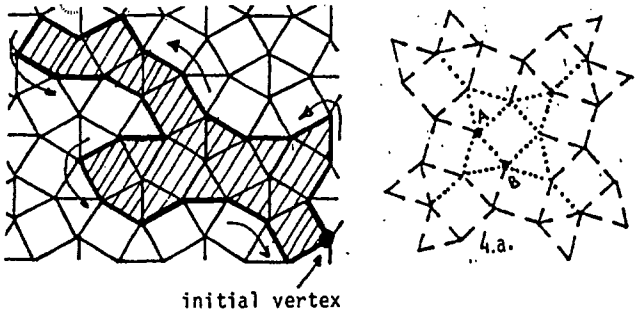
4-case: $m_1G|n_1G||$ then $m_1m_2...m_t\alpha 0m_2'...m_t'$ and $n_1n_2...n_t\alpha 0n_2'...n_t'$
$\sphericalangle(0,m_2')=\sphericalangle(1,n_2')\Rightarrow m_2'=1$ and $n_2'=5$ continuing we have $0m_2'...m_t'=0\,1\,5\,4$ and $1n_2'...n_t'=1\,5\,4\,0$, but $0\,1\,5\,4\,\alpha\,1\,5\,4\,0$

5-case: $m_1G|$, $n_1G|||$ then $m_1m_2...m_t\alpha 0m_2'...m_t'$ and $n_1n_2...n_t\alpha 2n_2'...n_t'$
$\sphericalangle(0,m_2')=\sphericalangle(2,n_2')\Rightarrow n_2'=0$ and $m_2'\in\{8,2\}$ if $m_2'=8$ applying σ_0 we have $m_1m_2...m_t\alpha 0\,2\,m_3''...m_t''$ $n_1n_2...n_t\alpha 2\,0\,n_2'...n_t'$
$\sphericalangle(0,m_3'')=\sphericalangle(2,n_3')\Rightarrow m_3''=0$ $\sphericalangle(2,m_3'')=\sphericalangle(0,n_3')\Rightarrow n_3'=2$
continuing we get $0\,2\,m_3''...m_t''=0\,2\,0\,2\,...$ and $2\,0\,n_3'...n_t'=2\,0\,2\,0\,...$ but this words are not from A'''.

6-case: $m_1G||$, $n_1G|||$ then $m_1m_2...m_t\alpha 1m_2'...m_t'$ and $n_1n_2...n_t\alpha 2n_2'...n_t'$
We are interested in the case when m_1 and n_1 do not satisfy any of previous observed cases. If for some i one of them is satisfied then we observe polygons $M_1M_{i+1}...M_{i-1}$ and $N_1N_{i+1}...N_{i-1}$ where $M_{t+k}=M_k$ and $N_{t+k}=N_k$. Since $\sphericalangle(1,m_2')=\sphericalangle(2,n_2')$ and m_2,n_2 do not satisfy cases 1,2,3,4,5 then $m_2'=2$ and $n_2'=9$. Next, we have $1m_2'...m_t'=1\,2\,9\,7$ and $2n_2'...n_t'=2\,9\,7\,1$ but $1\,2\,0\,7\quad 2\,9\,7\,1$ therefore $m_1m_2...m_t\alpha n_1n_2...n_t$.

LEMMA 8: Let G be the grid obtained by tessellations $3^2.4.3.4$. Then: if A and B are congruent polygons then $r(A)\alpha r(B)$.

PROOF: Equivalence classes with respect to relation ρ are:
$|=\{0,7,10,17\}$ $||=\{1,3,6,8,12,14,15,19\}$
$|||=\{2,4,5,9,11,13,16,18\}$ (see fig.4)



initial vertex

f(P) = 13 12 7 15 10 9 13 9 13 16 8 14 19 0
18 11 10 16 11 18 6 0 18 11 5

VECTORS: 10 11...18 19 are opposite
for : 0 1... 8 9

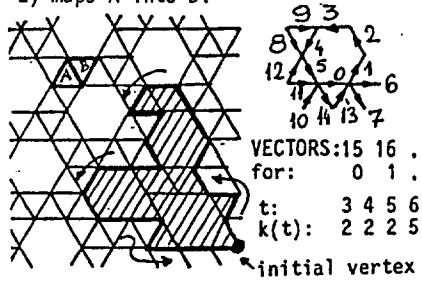
t: 3 4 5 6 7 8 9 10
k(t): 1 2 1 3 6 17 35 101

Fig. 4.

Let $M=M_1M_2...M_t$ and $N=N_1N_2...N_t$ are congruent polygons and $r(M)=m_1m_2...m_t$ and $r(N)=n_1n_2...n_t$. Case of interest is:

case: $m_1G||$, $n_1G|||$ then $m_1m_2...m_t\alpha 1m_2'...m_t'$ and $n_1n_2...n_t\alpha 11n_2'...n_t'$. Let us observe polygons $M'=ABM_3'...M_t'$ and $N'=BAN_3'...N_t'$ (see fig.4a):
$(r(M')=1m_2'...m_t'$ and $r(N')=11n_2'...n_t')$. Since M' and N' are congruent, there exist isometry S which maps plane into itself such that $S(M')=S(ABM_3'...M_t')=S(A)S(B)S(M_3')...S(M_t')=BAN_3'...N_t'=N'$. But then S is either
1) reflection in line s which is symmetry axes of segment $|AB|$.
or 2) half turn with centre in middle of segment $|AB|$.
If S is reflection then images of edges denoted by broken line (---) do not belong to grid G; therefore edges of polygon $BAN_3'...N'$ can be some of edges denoted with ····. Since polygonis conected, we conclude $n\leq 7$. But for $n\leq 7$ there are thirteen different α-equivalence classes and representatives of this classes are not congruent polygons so statement follows. In the case when S is half turn, proof is analogous.

For grid G obtained by tessellation $3^4.6$ (see fig.5) words which correspond to congruence polygons do not have to be α-equivalent. For example: for congruent triangles A and B (as it is shown in fig. 5) $r(A)\alpha r(B)$ but there is no isometry which 1) maps grid G into itself, 2) maps A into B.



VECTORS: 15 16 ... 28 29 are opposite
for: 0 1 ... 13 14

t: 3 4 5 6 7 8
k(t): 2 2 2 5 5 13

initial vertex

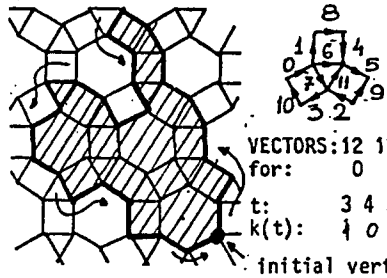
f(P) = 23 10 26 22 1 2 22 15 25 9 4 5 26 3 4
5 0 6 27 11 0 6

Fig. 5.

The proofs of following lemmas are omited, since they are analogous to proofs of Theorem 1 and Theorem 2.

LEMMA 9: Let G be the grid obtained by tesselations 3.4.6.4. Then: if M and N are congruent polygons in grid G then $r(M)\alpha r(N)$.

PROOF: Equivalence classes with respect to relation ρ are:
$|=\{0,1,2,3,4,5,12,13,14,15,16,17\}$
$||=\{6,7,8,9,10,11,18,19,20,21,22,23\}$



VECTORS: 12 13 ... 22 23 are opposite
for: 0 1 ... 10 11

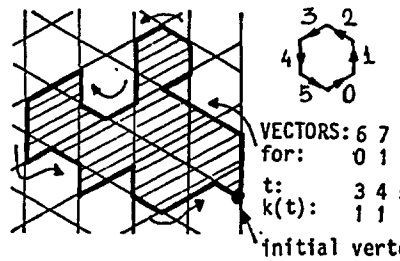
t: 3 4 5 6 7 8 9 10
k(t): 1 0 0 11 3 3 4

initial vertex

f(P) = 1 9 17 16 10 17 16 10 17 23 14 13 7 3
10 17 18 7 3 4 5 22 15 14 13 6 5 0

LEMMA 10: Let G be the grid obtained by tesselation 3.6.3.6. Then: if M and N are polygons in grid G then $r(M)\alpha r(N)$.

PROOF: There exist only one equivalence classes with respect to relation ρ.
$|=\{0,1,2,3,4,5,6,7,8,9,10,11\}$



VECTORS: 6 7 8 9 10 11 are opposite
for: 0 1 2 3 4 5

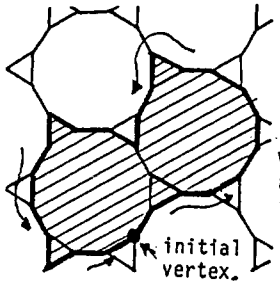
t: 3 4 5 6 7 8 9
k(t): 1 1 1 2 1 2 4

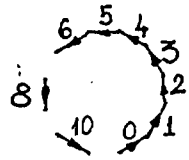
initial vertex

f(P) = 10 1 2 11 2 0 1 2 3 4 7 6 11 10 6 3 4
7 9 8 7 9 0 4 5 0 9 8

LEMMA 11: Let G be the grid obtained by tessellations 3.12^2. Then if M and N are congruent polygons in grid G then r(M)αr(N).

PROOF: Equivalence classes with respect to relation ρ are:
|={0,2,4,5,8,10,12,13,14,15,16,17}
||={1,3,5,7,9,11}.





VECTORS: 17 7 16 9 12 11 14 13 15
are opposite
for : 0 1 2 3 4 5 6 8 10

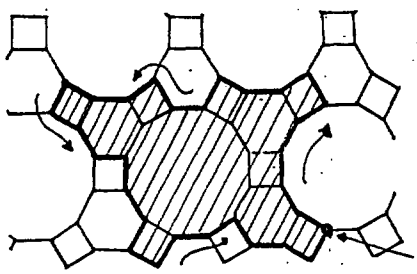
t: 3 12 13 14 15 16 17 18
k(t): 1 1 1 3 3 3 1 1

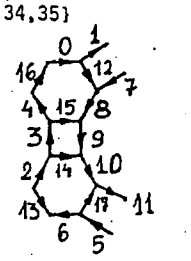
f(P) = 1 14 11 0 1 2 3 4 5 15 16 7 8 4 5 15
16 7 8 9 16 14 11 0

Fig. 8.

LEMMA 12: Let G be the grid obtained by tessellation 4.6.12 then: if M and N are congruent polygons in grid G then r(M)αr(N).

PROOF: Equivalence classes with respect to rellation ρ are:
|={0,2,4,6,8,10,18,20,22,24,26,28}
||={1,3,5,7,9,11,19,21,23,25,27,29}
|||={12,13,14,15.16,17,30,31,32,33,34,35}





initial vertex

f(P) = 29 28 27 26 25 31 19 18 29 35 6 31 19
18 29 35 23 22 15 9 10 35 23 16 0 1
13 24 23 16

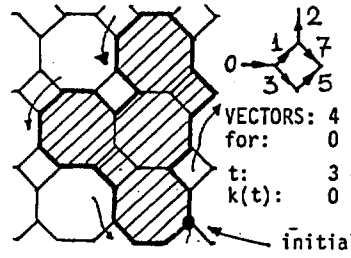
VECTORS: 18 19 ... 34 35 are opposite
for: 0 1 ... 16 17

t: 4 5 6 7 8 9 10 11 12 13 14 15 16
k(t): 1 0 1 0 1 0 1 0 3 0 2 0 9

Fig. 9.

LEMMA 13. Let G be the grid obtained by tessellation 4.8^2 then: if M, N are congruent polygons in grid G then r(M)αr(N).

PROOF: Equivalence classes with respect relation ρ are:
|={0,2,4,6} ; ||={1,3,5,7,9,10,11}



VECTORS: 4 11 6 8 9 10 are opposite
for: 0 1 2 3 5 7

t: 3 4 5 6 7 8 9 10 11 12 13 14
k(t): 0 1 0 0 0 1 0 1 0 2 0 4

f(P) = 2 8 1 2 5 10 2 8 4 9 6 7 9 8 4 9 6 7
0 3 6 7 0 1

Fig. 10.

Let grid G be obtained by one of semi-regular tessellations $3^3.4^2$, $3^2.4.3.4$, 3.4.6.4, 3.6.3.6, 3.12^2, 4.6.12, 4.8^2 then from lemmas 6-13 follows:

THEOREM 1: Circuits C_1 and C_2 in grid G are isomorphic circuits iff they are congruent circuits.

REFERENCES

|1| Doroslovački R., Stojmenović I., Tošić R., "Generating and counting triangular systems", BIT, 1987, 27, I, 1987, 18-24.

|2| Robert A. Metler, "Tessellation graph characterization using Rosettas", Pattern Recogrition Letters 4 (1986) 79-85

|3| Stojmenović I., Tošić R., Doroslovački R., "An algorithm for generating and counting hexagonal systems", Proc. of the 6-th Yugoslav Seminar on Graph Theory and Lectures for Research Seminar, Dubrovnik 1986, Institut of Mathematics, Univ. of Novi Sad, 1986, 189-198.

|4| Tepavčević A., Stojmenović I., "Counting Nonisomorphic pats in triangle-hexagonal grids", IX medjunarodni simpozij "Kompjuter na sveučilištu", 1987, IISOI, 1-4.

|5| Tošić R., Doroslovački R., "Characterization of hexagonal systems", Rev. of Res., Fac. of Sci. Math, Ser., Novi Sad 14, 2, 1984, 201-208.

|6| Zunić J., Stojmenović I., "Counting nonisomorpphic circuits in grids obtained by regular and semi-regular tessellations", XI medjunarodni simpozijum "Kompjuter na sveučilištu", 1989, to appear.

ON THE INTERSECTION OF TWO CONVEX POLYGONS

INFORMATICA 4/89

Keywords: convex polygon intersection

Dragan M. Acketa, Violeta Hauk and Dušan Surla
Institute of Mathematics, Novi Sad

Abstract. Given two convex polygons P and Q in the plane of size nP and nQ respectively, an O(nP+nQ) algorithm for determination of their intersection polygon was given in [5]. We elaborate the details and special cases of this algorithm (each of the 52 cases is recognized by using three very elementary boolean functions). This is incorporated within an implementation in Pascal language. It is shown (what makes a difference from the approach in [5]) that the generation of intersecting points may be separated from the construction of the intersection polygon itself.

O PRESEKU DVA KONVEKSNA POLIGONA. Neka su u ravni data dva konveksna poligona P i Q, sa brojem temena nP i nQ respektivno. U radu [5] je dat algoritam za odredjivanje njihovog presečnog poligona složenosti O(nP+nQ). U ovom radu su detaljno razradjeni svi mogući slučajevi tog algoritma (svaki od 52 slučaja se prepoznaje pomoću tri jednostavne logičke funkcije). Data je implementacija ovog algoritma u Pascal-u, u kojoj je, za razliku od [5], ukazano na mogućnost da se generisanje presečnih tačaka razdvoji od konstrukcije samog presečnog poligona.

INTRODUCTION

A branch of computational geometry is concerned with the problems related to the intersection of given geometric objects. Two problems should be distinguished here : determination and detection of the intersection. Shamos and Hoey have shown in [7] that $O[N^2]$ are necessary to determine the intersection of all pairs of the given N segments. They have given an O(NlogN) algorithm for testing the intersection of two segments and have applied it for testing the intersection of two simple polygons. They have obtained an O(NlogN) algorithm for detection of the intersection of N half-planes and have shown that the simplex method is not optimal. Plane-sweep algorithms for determining the intersection of geometric figures in the plane were considered by Nievergelt and Preparata ([4]). The idea of these algorithms was applied by Hertel et al. ([2]) for determination of the intersection of convex polyhedrons. Mehlhorn and Simos have shown in [3] that the computational complexity of determination of the intersection of polyhedra P and Q, one of which is convex, is of the form $O(n + m + s) * \log(n + m + s)$, where m and n respectively denote the number of edges of P and Q, while s denotes the number of edges of their intersection. Dobkin and Kirkpatrick have developped in [1] a method for testing the intersection of polyhedra with computational complexity $O(\log^2 N)$. Finally, O'Rourke et al. have obtained ([5]) an O(nP+nQ) algorithm for determining the intersection polygon of convex polygons P and Q with nP and nQ vertices respectively. This algorithm was also described in the monograph [6], Section 7.2.1.
It is this last algorithm that our paper is devoted to.

Let P and Q be two convex polygons in the plane and let P[i], 1<= i<= nP and Q[j], 1 <= j<= nQ respectively be their vertices (ordered w.r.t. the positive orientation, so that the polygon surfaces are placed on the left w.r.t. the oriented edges). In most cases the picture composed of both polygons is surrounded by altenatively placed "sickles" between some two neighbouring intersection points.

The algorithm from [5] can be sketched as follows :

```
i := 2;  j := 2;  k := 1;
REPEAT
    IF the edges P[i-1]P[i] and Q[j-1]Q[j]
    intersect   THEN   record intersection ;
    ADVANCE ;
    (* this procedure call increments either i
       or j by 1. The main idea is not to
       advance on the boundary of the
       polygon, the current edge of which
       contains a yet to be found
       intersection. The procedure ADVANCE
       also includes recording the vertices
       of P and Q, which should belong to
       the intersection polygon    *)
        k := k + 1
UNTIL k = 2 * (nP + nQ) ;
```

Roughly, the direction of advancing can be in non-degenerate cases determined by using the following two principles:
1) the principle of "forthcoming head advancing" : if one of the two current edges has already crossed over (or at least has reached) the line determined by the other , in front of that other edge, then that other edge, which is forthcoming, should advance.

2) the principle of "right head advancing" : if the principle 1) (which has the priority) cannot be applied, then the edge which should advance lies in the right-hand plane w.r.t to the opposite (oriented) edge.

It is guaranteed that two tours around the polygons are sufficient to "catch" all the intersection points. The exit from the REPEAT loop can be declared to be earlier in some cases. The case when no intersection point is recorded is specially treated in [5].

The application of convexity of the polygons is contained in the following observation : one can go around the boundary of a convex polygon (in the positive orientation) by keeping left at each vertex (that is, each edge lies in the left hyperplane with respect to the previous oriented edge). This way of moving around the polygons provides an easy possibility to determine which one of the two current edges should be followed when looking for the intersection points.

Our implementation of the above algorithm can be divided into the following two main steps :

STEP 1. Given the arrays P and Q of vertices of input convex polygons, construct the array R of their intersection points. (The main advancing idea is incorporated in this step)

STEP 2. Given the array R of the intersection points, construct the array S of vertices of the intersection polygon. (This step is a supplement to the basic algorithm)

Remarks. The points in the auxiliary array R are equipped with some additional data, which make the independence performance of the Step 2 possible. The original algorithm in [5] did not use such an auxiliary array, we have introduced it to make the performance of the algorithm more clear.

We make the following preparation for the possible second tour around the polygons:

```
FOR  i :=  1 to nP  DO   P[nP+i] := P[i];
FOR  j :=  1 to nQ  DO   Q[nQ+j] := P[j];
```

We shall use some abbreviations when explaining the advancing mechanism. The endpoints of the current two edges, P[i-1], P[i], Q[j-1] and Q[j] - will be denoted by A,B,C,D respectively.

We proceed with the complete Pascal code of our implementation, excluding some obvious abbreviations. The program is accompanied by appropriate commentaries .

P R O G R A M

```
PROGRAM Advance;

CONST
  co = 50;
       (* max.number of points in a polygon *)
 eps = 0.001; (* tolerance for comparing reals
                 and for testing parallelism *)
TYPE
  realpoint = RECORD  x, y : real  END;
  seq = array[1..co] OF realpoint;
  PQ = 'P'..'Q';
  intersection_point = RECORD
                          point : realpoint;
                          i, j : integer;
                          exit : PQ
                       END;
  interseq = array[1..co] OF intersection_point

(*     Introducing of special data type for
recording intersection points is essential,
since it allows an independent performance of
Step 2 in our algorithm. If an intersection
point is determined by the edges   P[i-1] P[i]
and  Q[j-1] Q[j] , then the indices  i  and  j
are equal to the corresponding components of
the record.  The record component  exit
denotes which one of the polygons  P  and  Q
should be followed on the boundary of their
intersection, starting from the current
intersection point  (the advance step will be
always made on the opposite polygon).  *)

VAR  P, Q, S :seq;   R :interseq;
 (* two input sequences P and Q, the output
    sequence S, and the auxiliary sequence R *)
    nP,  nQ, nR, nS,
 (* the  cardinalities  of the sequences *)
    k :integer;
    jump   :boolean;   (* may we jump over
         the procedure Construct_polygon  ? *)
 (* We start with seven routines needed to
distinguish between the cases which decide the
actions during advancing :  *)

FUNCTION  Eq( A, B :realpoint) : boolean;
(* Are the points A and B approximately equal ?
*) BEGIN Eq := (abs(sqr(A.x - B.x) +
                   sqr(A.y - B.y)) < eps) END;

FUNCTION  Eqc(a, b :real) : boolean;
  (* Are the real numbers  a  and  b
     approximately equal ? *)
  BEGIN   Eqc := (abs(a - b) < eps)  END;

FUNCTION Coef( A, B: realpoint ): real;
  (* Coefficient of the line determined by the
     points A and B *)
  BEGIN   Coef := (B.y - A.y)/(B.x - A.x)  END;

PROCEDURE Line_intersection ( A, B, C, D :
     realpoint;  VAR intersection: realpoint );
  (* determines the intersection of the lines
     AB  and  CD *)
  BEGIN
    IF NOT Eqc(A.x, B.x) AND NOT Eqc(C.x, D.x)
    THEN  BEGIN (* none of the lines AB and CD
                   is  parallel to y-axis *)
    intersection.x :=  (Coef(A,B)*A.x - A.y -
                        Coef(C,D)*C.x + C.y) /
                       (Coef(A,B) - Coef(C,D));
    intersection.y :=
       Coef(A,B)*(intersection.x-A.x)+A.y; END
    ELSE
    (* if AB is "vertical" , but CD is not  *)
    IF  Eqc(A.x,B.x) AND  NOT Eqc(C.x,D.x)
    THEN  BEGIN
    intersection.x := A.x;
    intersection.y := Coef(C,D)
       *  (intersection.x - D.x)+D.y;  END
    ELSE  BEGIN
    (* if CD is "vertical" ,but AB is not  *)
    intersection.x := C.x;
    intersection.y := Coef(A,B)
       *  (intersection.x - B.x)+B.y;  END;
  END;   (* Line_intersection *)

FUNCTION Paral( A,B,C,D :realpoint ) :boolean;
  (* Are the lines AB and CD approximately
     parallel ? *)     VAR  par :integer;
  BEGIN
    IF NOT Eqc(B.x, A.x) AND
       NOT Eqc(C.x, D.x) THEN  par := 1;
    IF     Eqc(B.x ,A.x) AND
       NOT Eqc(D.x ,C.x) THEN  par := 2;
    IF NOT Eqc(B.x ,A.x) AND
           Eqc(D.x ,C.x) THEN  par := 3;
    IF     Eqc(B.x ,A.x) AND
           Eqc(D.x ,C.x) THEN  par := 4;
    (* we must avoid comparing some infinite or
extremely large coefficients ; this is  the
reason why we treat separately the cases when
at  least  one  of  the  considered  line
segments is vertical *)
```

```
 CASE   par   OF
 1: IF NOT Eqc(Coef(A,B), Coef(C,D)) THEN BEGIN
Line_intersection ( A, B, C, D, intersection );
Paral :=  ( abs(intersection.x) > (1/eps) )  OR
          ( abs(intersection.y) > (1/eps) ) END
    ELSE   Paral := TRUE;
 2,3: BEGIN
Line_intersection ( A, B, C, D, intersection );
Paral := ( abs(intersection.y) > (1/eps) ) END;
 4: Paral := TRUE;
 END  (* CASE  par *)
 END; (* Paral *)

PROCEDURE   Midpoint( F, G, H: realpoint;
                      VAR  mid:realpoint );
  (* Generates the middle one among three
collinear points. The procedure is never
applied to non-collinear points *)
  BEGIN     IF  F.x <> G.x  THEN  BEGIN
     IF  (F.x < G.x)  AND  (G.x < H.x)  THEN
         mid := G;
   (*  analogously  FOR  the  remaining  5
       permutations of {F,G,H} *)  .....   END
     ELSE  BEGIN
(* the "vertical" case is treated separately *)
     IF  (F.y < G.y)  AND  (G.y < H.y)  THEN
         mid := G;
   (*  analogously  FOR  the  remaining  5
       permutations of {F,G,H} *)
        .....  END
  END;  (* Midpoint *)

FUNCTION   M( F, G, H :realpoint ): boolean;
  (* TRUE iff G is the middle one among
distinct  collinear  points  F, G, H.
This function is never applied to non-
collinear points  *)
   VAR   mid: realpoint;
   BEGIN
            M := FALSE;
            IF  (NOT Eq(F,G)) AND (NOT Eq(F,H))
            AND (NOT Eq(G,H)) THEN BEGIN
                Midpoint( F, G, H, mid );
                M := Eq(G, mid)
            END;
   END;

(* In addition, we give two standard functions,
the second of  which is used for the actions
during the advancing process, as  well as for
recognizing the relationship between the two
input  polygons in the cases when less than
two intersection points  are found: *)

FUNCTION  Vector_product(A,B,C:realpoint):real;
  BEGIN   Vector_product := A.x * (B.y - C.y) +
    B.x * (C.y - A.y) +  C.x * (A.y - B.y) END;

FUNCTION Right (Tail, Head, Point: realpoint ):
  boolean; (*  TRUE  iff  Point lies  in the
Right-hand  half-plane  with respect to the
oriented vector from Tail to Head  *)
 BEGIN  Right :=
 (Vector_product( Tail, Head, Point ) < 0) END;

  (* The following procedure plays a central
      role in our paper *)

PROCEDURE Extract_case(A,B,C,D:realpoint;
          VAR  E: realpoint; VAR cas:integer);
  (*  outputs the cases which are used for
making decisions  during the advancing
process within Step 1 *)
    BEGIN  (* Extract_case *)
(*  If the supporting lines of edges AB and CD
coincide,  then  there  are  26  distinct
possibilities. Namely, when the points  A and
B are fixed (assume that A < B in some
ordering), then there are  5  essentially
different possibilities for the location of the
point  C:  C<A , A<C<B , B<C , C=A , C=B.
Similarly there are  6 further possibilities
for the location of the point D with each of
the  first  three  cases,  as  well  as  4
possibilities with each of the last two cases.
```

Now assume that the supporting lines of the edges AB and CD intersect in the (reachable) point E. If the point E does not belong to the set {A,B,C,D} ,then there are 9 possibilities, depending on which of the points A, B, E, respectively of the points C, D, E - is the middle one. Similarly, if the point E coincides with exactly one of the points A, B, C, D, then with each one of the 4 cases there are 3 possibilities depending on the middle point on the opposite line. Finally, we should add the possibilities E=A=C , E=A=D , E=B=C , E=B=D . This makes the total of 25 cases with the reachable intersection of the two lines.

If the current two edges are parallel or close to parallel (small angle deviations up to a given bound are allowed), then we make difference depending on whether the supporting lines coincide or not. In the second case, it is obvious that the intersection point of the lines (which is either infinitely far or very far away) cannot be a vertex of the intersection polygon.

The global scheme for deciding which of the two edges AB and CD should advance seems as follows:

```
Let  p  and  q  denote the supporting lines
of the edges  AB  and  CD  respectively.
IF the lines p and q are (almost) parallel THEN
   IF the lines  p  and  q  (almost) coincide
      THEN  goto  cases  1 - 26
   ELSE goto case  27
ELSE  determine the intersection  E  of the
lines  p  and  q  and goto cases  28 - 52 . *)

IF Paral( A, B, C, D) THEN
    IF Vector_product( A, B, D ) = 0 THEN BEGIN
      (* the coincidence of supporting lines *)
      IF M(B,A,C) AND M(A,C,D)  THEN cas :=  1;
  ..................................... (*
 1:M(B,A,C),M(A,C,D)      2:M(B,A,C),M(A,D,C)
 3:M(B,A,C), Eq(A,D)      4:M(B,A,C),M(A,D,B)
 5:M(B,A,C), Eq(B,D)      6:M(B,A,C),M(A,B,D)
 7: Eq(A,C),M(B,A,D)      8: Eq(A,C),M(A,D,B)
 9: Eq(A,C), Eq(B,D)     10: Eq(A,C),M(A,B,D)
11:M(A,C,B),M(B,A,D)     12:M(A,C,B), Eq(A,D)
13:M(A,C,B),M(A,D,C)     14:M(A,C,B),M(B,D,C)
15:M(A,C,B), Eq(B,D)     16:M(A,C,B),M(A,B,D)
17: Eq(B,C),M(B,A,D)     18: Eq(B,C), Eq(A,D)
19: Eq(B,C),M(A,D,B)     20: Eq(B,C),M(A,B,D)
21:M(A,B,C),M(B,A,D)     22:M(A,B,C), Eq(A,D)
23:M(A,B,C),M(A,D,B)     24:M(A,B,C), Eq(B,D)
25:M(A,B,C),M(B,D,C)     26:M(A,B,C),M(A,C,D)
  *) .....................................
      IF M(A,B,C) AND M(A,C,D)  THEN cas := 26;
END
ELSE   (* IF the supporting lines are
     parallel but NOT coincident *) cas :=  27
  ELSE (* IF NOT Paral *)  BEGIN
      Line_intersection ( A,B,C,D,E );
      IF M(B,A,E) AND M(D,C,E)  THEN cas := 28;
  ..................................... (*
28:M(B,A,E),M(D,C,E)     29:M(B,A,E),M(C,D,E)
30:M(B,A,E),M(C,E,D)     31:M(A,B,E),M(D,C,E)
32:M(A,B,E),M(C,D,E)     33:M(A,B,E),M(C,E,D)
34:M(A,E,B),M(D,C,E)     35:M(A,E,B),M(C,D,E)
36:M(A,E,B),M(C,E,D)     37: Eq(A,E),M(D,C,E)
38: Eq(A,E),M(C,D,E)     39: Eq(A,E),M(C,E,D)
40: Eq(B,E),M(D,C,E)     41: Eq(B,E),M(C,D,E)
42: Eq(B,E),M(C,E,D)     43: Eq(C,E),M(B,A,E)
44: Eq(C,E),M(A,B,E)     45: Eq(C,E),M(A,E,B)
46: Eq(D,E),M(B,A,E)     47: Eq(D,E),M(A,B,E)
48: Eq(D,E),M(A,E,B)     49: Eq(A,E), Eq(C,E)
50: Eq(A,E), Eq(D,E)     51: Eq(B,E), Eq(C,E)
52: Eq(B,E), Eq(D,E)  *)  .....................
      IF  Eq(B,E) AND  Eq(D,E)  THEN cas := 52;
  END (* IF NOT Paral *)
END; (* Extract_case *)


(* Note that all the cases are determined by
calling only four elementary functions : Eq, M,
Paral and Vector_product *)
```

```
PROCEDURE Intersection_points ( nP, nQ:integer;
P, Q: seq; VAR nR, nS:integer;  VAR R:interseq;
VAR S:seq; VAR jump: boolean );
 (* This is the central procedure of the
algorithm. It completes      the Step 1 by an
iterative advancing . Extracting the cases
(on the basis of the previous procedure) ,
accompanied by the      corresponding actions ,
are performed along the way  *)
  VAR  A,B,C,D,E:realpoint; i,j,k,cas: integer;
       stop :boolean; (* should we exit from
                         the main loop? *)

   (* The following eight procedures (which are
interior and use  the side effect w.r.t. the
procedure Intersection_points)  are used for
performing  different advancing actions
depending on the case, which is extracted: *)


PROCEDURE   Record point   ( Point: realpoint;
                             Exit: PQ );
   (* registers the data on the current point,
      which are required for the "enriched"
      array  R  *)       VAR  okay: boolean;
      BEGIN     okay := FALSE;
      IF  nR > 0  THEN
          okay := NOT Eq( Point, R[nR].point );
   (* if  the current intersection (E) is
equal  to  the previous one  (R[nR]), then we
should not save it *)
      IF  (nR = 0) OR okay  THEN          BEGIN
   nR := nR + 1;   R[nR].point  :=  Point; END;
   (* the three additional components of the
record should  be always updated, regardless of
whether we are  actually advancing around the
intersection polygon: *)
   R[nR].i := i mod nP;
   IF R[nR].i = 0 THEN R[nR].i := nP;
   R[nR].j := j mod nQ;
   IF R[nR].j = 0 THEN R[nR].j := nQ;
   R[nR].exit := Exit;  END; (* Record_point *)


PROCEDURE Go_on ( Exit: PQ );
   (* performs the elementary advancing step in
most of the cases when there exists the
intersection of the two supporting       lines.
If the intersection belongs to the both of the
current edges, then the procedure writes it
down *)   BEGIN
  IF  cas  IN  [36,39,42,45,48,49,50,51,52]
   (* equivalently, IF both the points defining
     cas  coincide  with  the intersection E *)
           THEN  Record_point (E, Exit);
  IF  Exit = 'Q'  THEN    i:= i + 1;
  IF  Exit = 'P'  THEN    j:= j + 1
   (* advance along P if the exit Q is recorded
      and conversely *)       END;  (* Go_on *)


PROCEDURE Treat (Tail, Head, Point: realpoint;
   Exit: PQ );  (*  materializes the principle
         of "right head advancing" *)   BEGIN
IF Right (Tail, Head, Point)  THEN Go_on (Exit)
ELSE   Go_on (chr(ord('P')+ord('Q')-ord(Exit)))
(* that is, use the opposite exit *)       END;


PROCEDURE Write_down_1 ( X:realpoint );
   (* records the point X, the exit (as well as
the advancing   direction) is   actually
irrelevant, since the next step will  have the
same outcome in the both possible cases *)
BEGIN    Record point( X, 'Q'); i:= i+1    END;


PROCEDURE Write_down_2 ( X, Y:realpoint );
   (*  similar to the previous procedure, but
records two  points, X and Y *)
   BEGIN    Record_point( X, 'Q' ) ;
            Record_point( Y, 'Q' ) ;
            i:= i+1                        END;


PROCEDURE Exit_0;  (* a pair of edges may be
sufficient  to  decide  that  the  polygon
intersection is empty *)
   BEGIN  stop := TRUE;  jump := TRUE;
          writeln(' Polygon surfaces of P AND Q
                    are disjoint')      END;

PROCEDURE Save_exit_1 ( X :realpoint );
 (* used if it is immediately recognized
    that  X  is the only intersection point *)
   BEGIN  stop := TRUE;  jump := TRUE;
          nS := 1;  S[1] := X;    END;

PROCEDURE Save_exit_2 ( X, Y :realpoint );
 (*  used if it is immediately recognized that
X and Y are the only two intersection points *)
   BEGIN  stop := TRUE;  jump := TRUE;
          nS := 2;  S[1] := X;  S[2] := Y;  END;

   BEGIN  (* Intersection_points *)
     jump := FALSE;  stop := FALSE;
     i := 2;   j := 2;   nR := 0;   k := 1;
 (* k = the number of passes through REPEAT *)

   REPEAT   (* the main loop, each pass
through  it  corresponds to one advance step *)

   IF i = 1 THEN  A := P[nP]  ELSE  A := P[i-1];
   IF i>nP  THEN  i := i MOD nP;    B := P[i];
   IF j = 1 THEN  C := Q[nQ]  ELSE  C := Q[j-1];
   IF j>nQ  THEN  j := j MOD nQ;    D := Q[j];
(* The above four IF-statements establish a
connection between the first and the second
tour  around the input polygons  *)
(*  Recognition of the mutual position of the
current two edges  AB and CD  *)
   Extract_case ( A, B, C, D, E, cas );
(*  Performing the appropriate action depending
on  the case involved *)
            CASE cas OF
              3 :   Write_down_1( A ) ;
            6,10 :  Write_down_2( A, B ) ;
          4,5,8,9 : Write_down_2( A, D ) ;
             20 :   Write_down_1( C ) ;
             16 :   Write_down_2( C, B ) ;
           14,15 :  Write_down_2( C, D ) ;
            1,25 :  Exit_0 ; (* disjoint P,Q *)
              7 :   Save_exit_1( A ) ;
             24 :   Save_exit_1( B ) ;
          21,22 :   Save_exit_2( A, B ) ;
   11,12,17,18 :    Save_exit_2( A, C ) ;
             23 :   Save_exit_2( B, D ) ;
          13,19 :   Save_exit_2( C, D ) ;
             27 : IF Right( A,B,C ) AND
                     Right( C,D,A ) THEN Exit_0
                     ELSE Treat( A,B,D,'P' );
             32 :   Treat( B,E,D,'P' );
28,30,34,36,37,39,43,45,49 :Treat( E,B,D,'P');
 2,31,33,40,42,44,47,51,52 :  i := i + 1;
   26,29,35,38,41,46,48,50 :  j := j + 1;
   (* Most of the cases in the last two groups
are solved by  using the principle of
"forthcoming head advancing"  *)
   (* Validity of the above actions is easily
checkable by hand. We suggest the  reader to
draw the small pictures  (containing just two
oriented edges each), which  correspond to each
one of the 52 cases *)
            END;  (* CASE cas OF *)

IF Eq( R[nR].point , R[1].point) AND ( nR > 1 )
THEN  stop := TRUE      (* the equality of
R[nR].point and R[1].point  means that a non-
trivial intersection  polygon is already
completed *)
ELSE  IF  Eq( R[nR].point  ,  R[2].point)  AND
( nR > 2 )  THEN  (* It may happen that we are
unable to stop by  recognizing the second
appearance of  R[1].point. One  call of the
procedure  Write_down_2     records   two
intersection points *)
      BEGIN  nR := nR - 1;  stop := TRUE; END
  ELSE  (* augment the number of passes by 1 *)
          k := k + 1
UNTIL (k = 2*(nQ+nP)) OR stop;
```

```
(* All the intersection points (if they exist
at all) will be discovered before the
boundaries of the input polygons are passed
twice. On the other hand, the construction of
the intersection polygon should be terminated
at the moment when its boundary becomes closed
or when it for some other reason becomes clear
that no other intersection points can be found
*)
        IF nR > 1 THEN   nR :=   nR - 1;
(* This is necessary because R[1].point is
counted twice *)

END;  (* Intersection_points *)

(* The last two procedures are used within the
Step 2. of our implementation *)


PROCEDURE  Construct_polygon (nP,nQ,nR:integer;
P,Q:seq; R:interseq; VAR nS:integer;VAR S:seq);
(* completes the intersection polygon in the
cases when at least two intersection points
are present, while no trivial final decision
can be made (consequently, the boolean
variable "jump" is FALSE) *)
    VAR  start, finish , (* the first and the
last (inclusively) index of the vertices (of an
input polygon) lying on the inner bound of
a considered sickle *)    w, y :integer;
BEGIN   (* appropriate initializations *)
     nS := 0;      R[nR+1]:=R[1];
   FOR  w:= 1 TO nP DO  P[nP+w] := P[w];
   FOR  w:= 1 TO nQ DO  Q[nQ+w] := Q[w];
(* The main FOR-loop; each pass through it
corresponds to the completion of such a part of
the boundary of the  ntersection  polygon,
which represents the interior boundary of a
sickle between the two input polygons *)
   FOR  w:= 1 TO nR  DO BEGIN
(*  Put  the  intersection  point  into  the
     intersection polygon *)
     nS := nS + 1;   S[nS] := R[w].point;
(* Put the "sickle" points of P or Q, which
should lie on the boundary of the intersection
polygon between  R[w].point and R[w+1].point *)
     IF  R[w].exit = 'P'  THEN BEGIN
         start  := R[w].i ;
         finish := R[w+1].i - 1;
(* we use the inner indices of the vertices of
the   inner  polygon  w.r.t.  the sickle  *)
         IF (R[w].i > R[w+1].i)  THEN
            finish := finish + nP;
(* the connection between two tours is
established *)
         IF  Eq(  R[w].point, P[start] ) THEN
                  start := start + 1;
         IF  Eq( R[w+1].point, P[finish] ) THEN
                  finish := finish - 1;
(* the last two statements are necessary in
order to  avoid duplication of the vertices of
the intersection polygon in degenerate cases,
when an  initial polygon vertex coincides with
an  intersection point *)
         FOR y := start  TO  finish  DO  BEGIN
             nS := nS + 1; s[nS] := P[y]   END
     END;  (* IF  R[w].exit = 'P' *)
     IF  R[w].exit = 'Q'  THEN BEGIN
(* Replace i, P, nP  in the part under
IF R[w].exit = 'P' by  j, Q, nQ  respectively
and repeat the rest *)
          .........                  END;
   END;  (* FOR  w:= 1 TO nR *)
END; (* Construct_polygon *)

(*  Remark.     Although  the  procedure
Construct_polygon includes double-nested  FOR-
loops, the required number of steps is not
greater than  O(nP + nQ). This follows from the
elementary fact that the maximal number of
vertices of the intersection polygon is equal
to nP + nQ. *)
```

```
(*  The following procedure deals with the
cases  nR = 0  and     nR = 1. In these cases
additional tests are performed in order     to
check whether the surface of one polygon is
completely included into the surface of the
other. (Note that similar dilemas are not
present whenever nR >= 2; two intersection
points are sufficient for proper initialization
of the tour around the intersection polygon).
These  additional   tests   do  not  spoil
linearity of the whole algorithm. For instance,
consider the     question whether the whole  P
is inside  Q. It suffices to     check whether
the point  P[1] (or the point P[2] if P[1]
coincides with the intersection point) is in
the interior of Q. This is true iff P[1] (P[2])
lies inside the left half-plane w.r.t. each
oriented edge of Q , that is, iff that point is
not possibly (placed to the) right from (an
edge of )  Q  *)


PROCEDURE   None_or_one( nP, nQ: integer;
     P, Q: seq;   VAR nS: integer;  VAR S: seq;
     VAR jump: boolean );
(* applies to the cases  nR = 0  and  nR = 1 *)
     VAR     i,j : integer;  pp,qq : realpoint;
             prfP , prfQ : boolean;
(* possibly right from P AND Q respectively *)
BEGIN
    jump := TRUE; prfP := FALSE; prfQ := FALSE;
    j := 1;   Q[nQ + 1] := Q[1];   pp := P[1] ;
    IF (nR = 1) AND Eq(P[1],R[1].point) THEN
       pp := P[2];
    REPEAT  j := j + 1;
       IF Right( Q[j-1], Q[j], pp ) THEN
          prfQ := TRUE
    UNTIL  ( j = nQ + 1 )  OR  prfQ;
    i := 1;   P[nP + 1] := P[1];   qq := Q[1]
    IF (nR = 1) AND Eq(Q[1],R[1].point) THEN
       qq := Q[2];
    REPEAT    i := i + 1;
       IF Right( P[i-1], P[i], qq ) THEN
          prfP := TRUE
    UNTIL  ( i = nP + 1 )  OR  prfP;
    IF NOT prfQ   THEN BEGIN
       nS := nP; S := P  END
       (* P is contained within Q *)
    ELSE
    IF NOT prfP   THEN BEGIN
       nS := nQ; S := Q  END
       (* Q is contained within P *)
    ELSE   (* the polygons   P   and   Q
       are   placed  outside  each  other *)
    IF  nR = 0   THEN                BEGIN
       writeln(' Polygon surfaces of P AND Q
       are disjoint ');  nS := 0      END
    ELSE  (* IF  nR =1 *)             BEGIN
       nS := 1 ;   S[1] := R[1].point  END
 END;  (* None_or_one *)


 BEGIN    (* Main program *)
   (* Input *)
   read(nP,nQ);
   FOR k:= 1 TO nP DO  BEGIN
       read(P[k].x);  read(P[k].y)   END;
   FOR k:= 1 TO nQ DO  BEGIN
       read(Q[k].x);  read(Q[k].y)   END;
   (*  Step 1 *)
   Intersection_points
       (nP,nQ,P,Q,nR,nS,R,S,jump)
   (*  Step 2 *)
   IF  nR < 2 THEN
      None_or_one (nP,nQ,P,Q,nS,S,jump );
   IF NOT jump THEN
      Construct_polygon (nP,nQ,nR,P,Q,R,nS,S);
   (* Output *)
   writeln(' The intersection polygon : ');
   FOR k:= 1 TO nS DO   BEGIN
      write(S[k].x,' '); writeln(S[k].y)  END
 END.
```

REFERENCES

[1] Dobkin, D.P., Kirkpatrick, D.G., Fast detection of polyhedral intersection, Theoretical Computer Science 27 (1983), 241-253.

[2] Hertel, S., Mantylla, M., Mehlhorn, K., Nievergelt, J., Space sweep solves intersection of convex polyhedra, Acta Informatica 21, 501-519, 1984.

[3] Mehlhorn, K., Simon, K., Intersection two polyhedra one of which is convex,

[4] Nievergelt, J., Preparata, F.P., Plane-sweep algorithms for intersecting geometric figures, Programming Techniques and Data Structures, Communications of the ACM, Vol. 25, No. 10, 739-747, 1982.

[5] O'Rourke, J., Chien, C.B., Olson, T., Naddor, D., A new linear algorithm for intersecting convex polygons, Computer graphics and Image processing, 384-391.

[6] Preparata, F.P., Shamos, M.I., Computational geometry, Springer-Verlag, 1985.

[7] Shamos, M.I., Hoey, D., Geometric intersection problems, Proc. 17th Annual IEEE Symp. on Foundations of Computer Science, 208-215, 1976.

FRAKTALI – ZNANOST ALI UMETNOST

INFORMATICA 4/89

Keywords: fractal, iterated function method, Julia set, Mandelbrot set

Jasna Đonlagić, Nikola Guid
Tehnična fakukteta Maribor

V naravi je veliko objektov in pojavov, kot n.pr. drevesa, gore, oblaki, pretok tekočin, rast populacije, oblika možganskih gub ter podobni pojavi, ki iz določenega reda preidejo v nered in ki jih ne moremo matematično predstaviti z običajnimi orodji, lahko pa jih s fraktalno geometrijo oz. fraktali. V članku obravnavamo naravne in geometrične fraktale, metodo iterativnih funkcijskih sistemov (IFS) in nakažemo uporabo formalnih jezikov. Na koncu izdelamo algoritme za naslednje vrste fraktalov: Juliovo množico, Mandelbrotovo množico in fraktale določene po metodi IFS.

ABSTRACT

For some natural objects like trees, mountains, clouds, flow of liquids, population growth, surface of brain etc., which turn from order into chaos, it is impossible to describe all mathematical data, but it is possible to present them with fractal geometry or fractals. The purpose of this article is to present natural and geometrical fractals, iterated function method (IFS), and use of the formal languages in the fractal theory. At the end we show some algorithms for creating the following fractals: Julia set, Mandelbrot set, and fractals produced by IFS method.

1. UVOD

Beseda fraktal je latinskega izvora (fractus = zlomljen), torej naj bi spominjala na lomljenje, drobljenje. S fraktalno teorijo se je pričel ukvarjati Benoit B. Mandelbrot 1980. leta [6], čeprav so matematične osnove za nastanek te teorije ustvarili že mnogo prej P. F. Verhulst, Gaston Julia, Pierre Fatou, Adrien Douady in drugi [7].

Fraktale delimo na dve osnovni skupini, in sicer na: naravne in geometrične.

Članek obsega opis naravnih in geometričnih fraktalov ter uporabo formalnih jezikov na področju fraktalne teorije. V povezavi z naravnimi fraktali smo opazovali Verhulstov dinamični proces ter Juliove in Mandelbrotove množice. Pri geometričnih fraktalih smo podali metodo IFS in opisali nekatera pravila za generiranje geometričnih fraktalov. Prav tako smo razvili tri algoritme za generiranje fraktalnih slik.

2. NARAVNI FRAKTALI

Naravne fraktale lahko zasledimo na področju geografije, biologije, biokemije in fizike, ko želimo z njimi upodobiti naravne pojave, kot so na primer poplave rek, pretok tekočin, obliko možganskih gub, vaskularni sistem, vreme itn.

Ideja za nastanek naravnih fraktalov izvira iz biologije, ko je P. F. Verhulst 1845. leta definiral zakon rasti populacije [7]. Zagovarjal je trditev, da lahko določena populacija narašča tako dolgo, dokler ne doseže svojega razsežnostnega maksimuma X. Če le tega prekorači, velikost populacije pade. Potrebno je bilo več kakor sto let, da so dokazali vse nejasnosti te trditve. Prišli so do zanimivih rezultatov :

Če je faktor rasti majhen, se velikost populacije uravnava sama po sebi in zmeraj ostaja znotraj optimalne vrednosti X. To pomeni, da populacija narašča, kadar je pod optimalno vrednostjo X, in pada, kadar je nad optimalno vrednostjo X. V primeru, da je faktor rasti velik, takšen, da se populacija poveča za več kakor 200 %, pa optimalne vrednosti X ni mogoče več doseči.

Porodi se vprašanje, ali je rast populacije sploh kdaj tako velika? Seveda! Človeške populacije ne naraščajo tako hitro, toda pri nekaterih insektih tak pojav ni nenavaden. Ugotovili so, da pri 245 % povečanju populacije nastanejo oscilacije okoli optimalne vrednosti X z velikostjo periode 2,4,8,16,..., dokler pri 257 % ne preidemo v zmedo.

Kako si razlagamo besedo zmeda? To preprosto pomeni, da delovanja sistema ne moremo več nadzorovati, da je ušla izpod naše kontrole.

Najzanimivejša točka Verhulstovega dinamičnega procesa pa ni zmeda sama po sebi, temveč dogodek, ki spremeni red v nered. Verhulst je to trditev matematično dokazal. Z x_0 označi začetno velikost populacije, z x_n pa velikost populacije po n letih. Jakost rasti populacije R definira relativni prirastek populacije na leto:

$$R = \frac{x_{n+1} - x_n}{x_n}$$

Nato uvede določene predpostavke:

- Populacija lahko naraste do določenega maksimuma X (maksimum normirajmo, tako da je X=1)
- R se naj spreminja glede na velikost populacije in naj bo linearna funkcija od r (r imenujemo parameter rasti in r>0): $R=r(1-x_n)$

Ob teh predpostavkah dinamični zakon rasti preide v naslednjo obliko:

$$x_{n+1} = f(x_n) = (1+r)x_n^2 - rx_n$$

Ločimo dve stanji, pri katerih populacija ne spreminja velikosti: $x_0 = 0$ in $x_0 = 1$. Zanima nas stabilnost oz. nestabilnost teh dveh stanj. Pri $x_0 = 0$ ne pride do rasti populacije, saj nimamo s čim začeti. To stanje smatramo za nestabilno, saj že pri $0<x_0<<1$ nastopi rast populacije:

$$x_1 \cong x_0 - rx_0$$

$$x_2 \cong x_1 - rx_1$$

...itn.

Zaporedje $x_0, x_1, x_2, \ldots$, narašča, dokler ne doseže velikosti 1. Ali je stanje $x_0 = 1$ stabilno? Da bi prišli do odgovora, opazujmo razliko $\delta_n = x_n - 1$! Dinamični zakon nam omogoča izračun x_{n+1}, torej lahko določimo tudi naslednji δ_{n+1}.

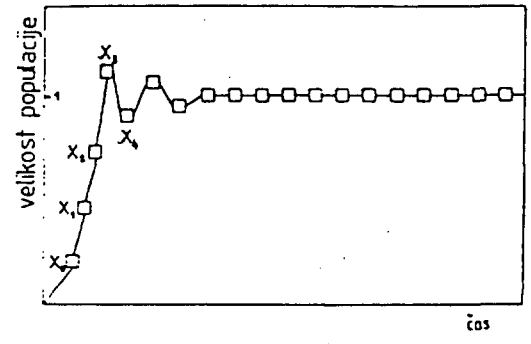
$$\delta_{n+1} \cong \delta_n(1-r)$$

Težimo za tem, da se x_{n+1} stabilizira pri 1, zato mora δ_{n+1} limitirati proti 0.

$$\lim_{n\to\infty} \delta_{n+1} = 0 \quad \Rightarrow \quad \lim_{n\to\infty} \delta_0(1-r)^n = 0$$

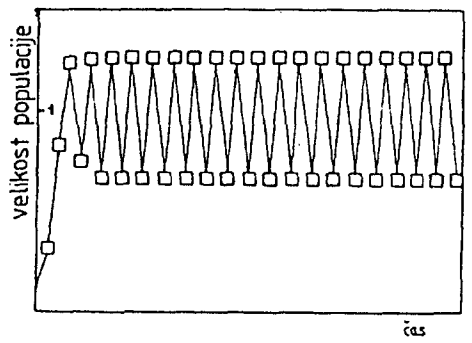
To bo veljalo, kadar bo $|\delta_{n+1}| < |\delta_n|$ (z x_{n+1} se bomo približevali k 1), če je $0<r<2$. Ali z drugimi besedami: Stanje $x_0 = 1$ je stabilno, če r leži v mejah med 0 in 2 ter nestabilno, če je r>2.

Oglejmo si primer, pri katerem postavimo $x_0 = 0.1$ in r=1.8 (slika 2.1).

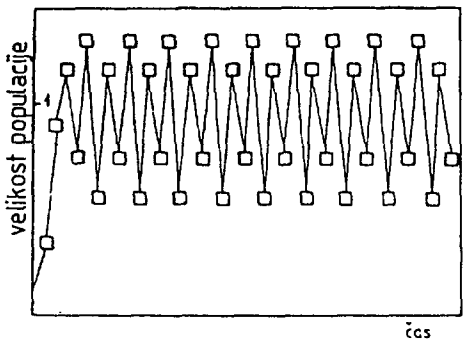


Slika 2.1 Rast populacije pri r=1.8 in $x_0 = 0.1$

Vidimo, da velikost populacije x sprva narašča, saj je pod maksimalno vrednostjo X=1. V četrtem koraku pride do prekoračitve. Zaradi tega se velikost populacije zmanjša in pade pod 1. To povzroči ponovno rast in takoj za tem ponoven padec itn., dokler se dokončno ne umiri pri stabilni vrednosti X=1. Zanimivejši položaj nastane, kadar je r>2. Zato raziščimo primer, ko postavimo $x_0 = 0.1$ in r=2.3 (slika 2.2) ter $x_0 = 0.1$ in r=2.5 (slika 2.3).

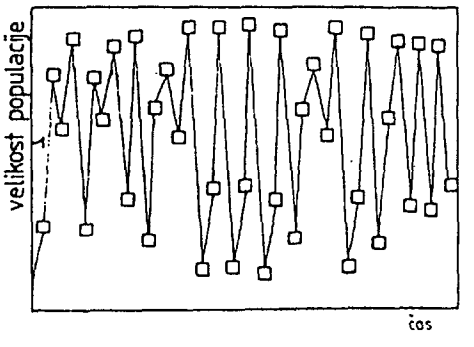


Slika 2.2 Rast populacije pri r=2.3 in x_0= 0.1



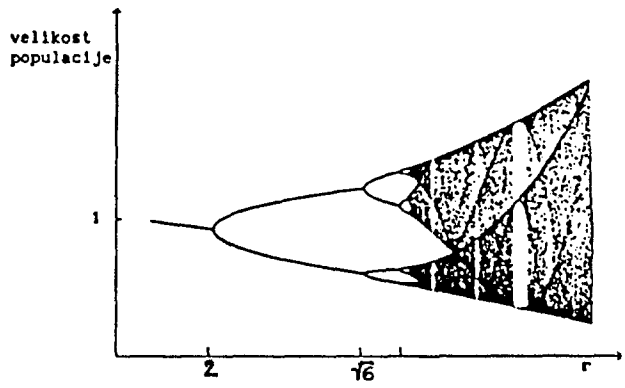
Slika 2.3 Rast populacije pri r=2.5 in x_0= 0.1

Ugotovimo, da pride do periodične oscilacije med nivoji. V prvem primeru dobimo periodo 2, v drugem pa 4. Če bi nadaljevali s postopkom, bi dobili periodo 8,16,32,..., dokler pri r=2.57 ne bi prešli v zmedo in periode ne bi bilo več mogoče določiti (slika 2.4).



Slika 2.4 Rast populacije pri r=2.57 in x_0= 0.1

V nadaljevanju si lahko zastavimo zanimivo vprašanje: Katero maksimalno vrednost lahko zavzame r, da bi dobili še enako stabilno oscilacijo (n.pr. s periodo 2)? Če izberemo $2<r<\sqrt{6}$, dobimo oscilacije s periodo 2, če pa izberemo r malo večji od $\sqrt{6}$, dobimo oscilacije s periodo 4 itd (slika 2.5).



Slika 2.5 Verhulstov dinamični proces

Vprašali se boste v kakšni povezavi je vse to skupaj s fraktali? Verhulstov proces je enodimenzionalen, Mandelbrot pa je raziskal popolnoma enak problem, le v 2D prostoru. Odločil se je opazovati kompleksna števila namesto realnih, pri čemer je sledil procesu $x_0, x_1, x_2, \ldots$, v ravnini, raje kakor na premici.

Dejal je, da imamo v ravnini **stekališča** (attractors), ki se "borijo" za svoj vpliv na ravnini. Točka x_0 se v dinamičnem procesu približuje enemu ali drugemu stekališču, lahko pa je na meji (ločnici) med dvema stekališčema in se ne more opredeliti za enega. Mandelbrotov proces je matematično ekvivalenten Verhulstovem procesu. Izhaja iz preproste formule:

$$x_{n+1} = f(x_n) = x_n^2 + c$$

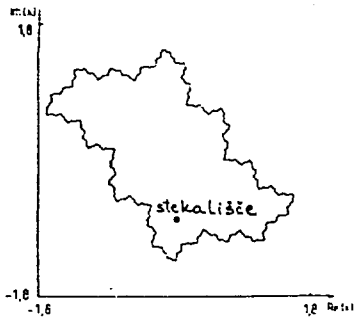
pri kateri ločimo dve situaciji:

1. Fiksiramo vrednost c in dovolimo, da x_0 zavzame različne vrednosti kompleksnih števil.
2. Fiksiramo x_0 in dovolimo spremembo c.

Pri prvi situaciji lahko postavimo c=0 ali c≠0.
a) Če je c=0 dobimo zaporedje iteracij $x_0, x_0^2, x_0^4, \ldots$, pri katerem ločimo tri različna stanja glede na vrednost x_0:

- Števila zaporedja postajajo vse manjša in manjša. Takšna situacija nastane, kadar je $|x_0| < 1$. Zaporedje limitira proti nič in pravimo, da je nič stekališče procesa $x \Rightarrow x^2$.
- Števila zaporedja postajajo vse večja in večja. Do tega primera pridemo, kadar je $|x_0| > 1$. Zaporedje limitira proti neskončnosti in pravimo, da je neskončnost stekališče procesa $x \Rightarrow x^2$.
- $|x_0| = 1$. V tem primeru so števila zaporedja na meji med dvema stekališčema. Mejo sestavlja krožnica enotskega kroga, ki razdeli ravnino na dve regiji. Nič predstavlja notranje stekališče, neskončnost pa zunanje stekališče.

b) Če predvidimo $c \neq 0$ dobimo zaporedje iteracij $x_0, x_1, x_2, \ldots$, pri katerem lahko prav tako nastanejo prej naštete možnosti. Razlika je le v tem, da notranje stekališče nima več vrednosti 0, temveč drugačno vrednost, in da meja ni več lepo ukrivljena, temveč je "nazobčana". Prav s tem pa smo porušili pravilnost geometrijskih likov in dobili lik, ki je podoben naravnemu. Imenujemo ga *fraktal* (slika 2.6).



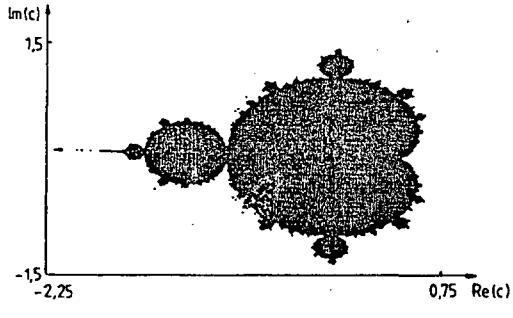
Slika 2.6 Fraktal

Pri podrobnejši analizi meje ugotovimo dve zanimivi lastnosti :

- Če vzamemo povečevalno steklo in pogledamo delček meje od blizu, ugotovimo, da je popolnoma enak, kakor če ga pogledamo brez povečevalnega stekla. To lastnost imenujemo *samopodobnost*.
- druga lastnost je *zrcalnost* .Če pogledamo obliko meje v enem kotu, ugotovimo njeno zrcalno sliko v nasprotnem kotu.

Fraktale, ki posedujejo takšne lastnosti, imenujemo **Julilove množice** (po francoskem matematiku Gaston Julii) [7].

Druga možnost je, da fiksiramo $x_0 = 0$ in dovolimo spremembo c po vsej kompleksni ravnini. Na tak način dobimo novo množico, ki jo imenujemo **Mandelbrotova množica M** (B. Mandelbrot je prvi objavil njen opis in jo prikazal na računalniku, slika 2.7).

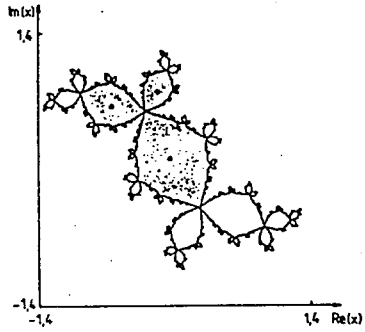


Slika 2.7 Mandelbrotova množica pri $x_0 = 0$

Zanimive so naslednje ugotovitve:

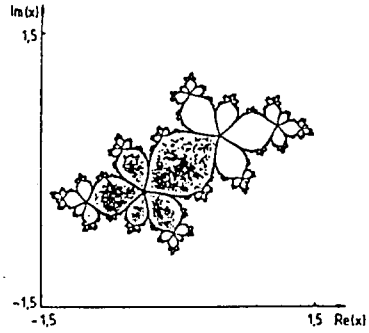
- Če izberemo c iz notranjosti glavnega dela telesa Mandelbrotove množice in ga po prvem postopku razvijemo v Juliovo množico, dobi Juliova množica obliko deformirane, nazobčane krožnice, ki obkroža stekališče (slika 2.6, podoben primeru 1 na sliki 2.13).

- Če izberemo c v notranjosti enega izmed "popkov" Mandelbrotove množice in ga razvijemo v Juliovo množico, bo Juliovo množico sestavljalo več deformiranih, nazobčanih krožnic, kjer vsaka izmed njih obkroža svoje stekališče (slika 2.8, primer 3 na sliki 2.13).



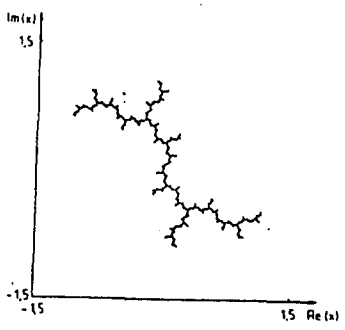
Slika 2.8 Primer fraktala, ki se nahaja v notranjosti enega izmed "popkov" Mandelbrotove množice

- Če izberemo c na meji, kjer se "popek" dotika glavnega dela telesa Mandelbrotove množice in ga razvijemo v Juliovo množico, dobimo t.i. parabolični primer. Zanj je značilno, da se več nazobčanih krožnic steka v eno točko (slika 2.9, primer 8 s slike 2.13).



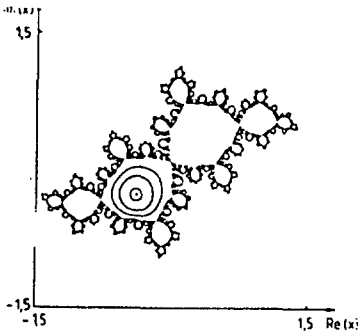
Slika 2.9 Parabolični primer fraktala

- Če izberemo c na "anteni" Mandelbrotove množice in ga razvijemo v Juliovo množico, dobimo fraktal imenovan *dendrit*. Zanj je značilno, da ima eno samo stekališče, ki je v neskončnosti. Zaradi tega tak fraktal nima notranjosti (slika 2.10, primer 4 na sliki 2.13).



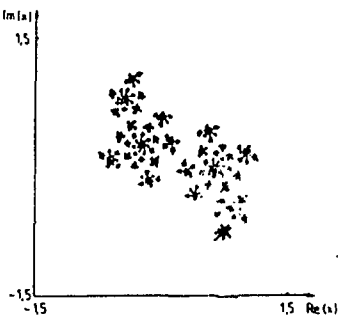
Slika 2.10 Primer dendrita

- Če izberemo c kjerkoli drugje na meji Mandelbrotove množice in ga razvijemo v Juliovo množico, pride do pojavov Siegelovih diskov. Če točka v iteracijskem postopku doseže Siegelov disk, bo v njem tudi ostala. To pomeni, da bo v krogu rotirala okoli stekališča, samega stekališča pa ne bo nikdar dosegla (slika 2.11, primer 9 s slike 2.13).



Slika 2.11 Primer fraktala s Siegelovimi diski

- Če izberemo c izven Mandelbrotove množice in ga razvijemo v Juliovo množico, bo Juliova množica nepovezana. To pomeni, da razpade v množico točk in tak fraktal imenujemo *Fatouov prah* (slika 2.12, primer 11 na sliki 2.13).



Slika 2.12 Fatouov prah

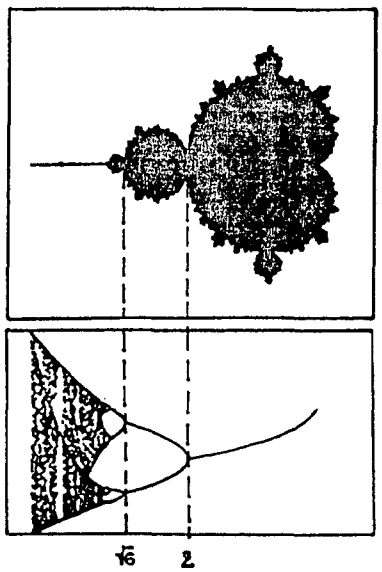
1983. leta je Dennis Sullivan dokazal, da so to edine oblike, ki jih Juliova množica lahko zavzame. Obstaja še ena možnost, tako imenovani Hermanov obroč (Herman ring), ki pa ne nastane v primeru $x \mapsto x^2$. Teoretično je dokazano, da lahko do njegovega pojava pride v drugih primerih, čeprav praktično še ni bil nikdar opazovan [7].

Slika 2.13 prikazuje nekaj Juliovih množic, ki jih dobimo z ustrezno izbiro parametra c iz Mandelbrotove množice.

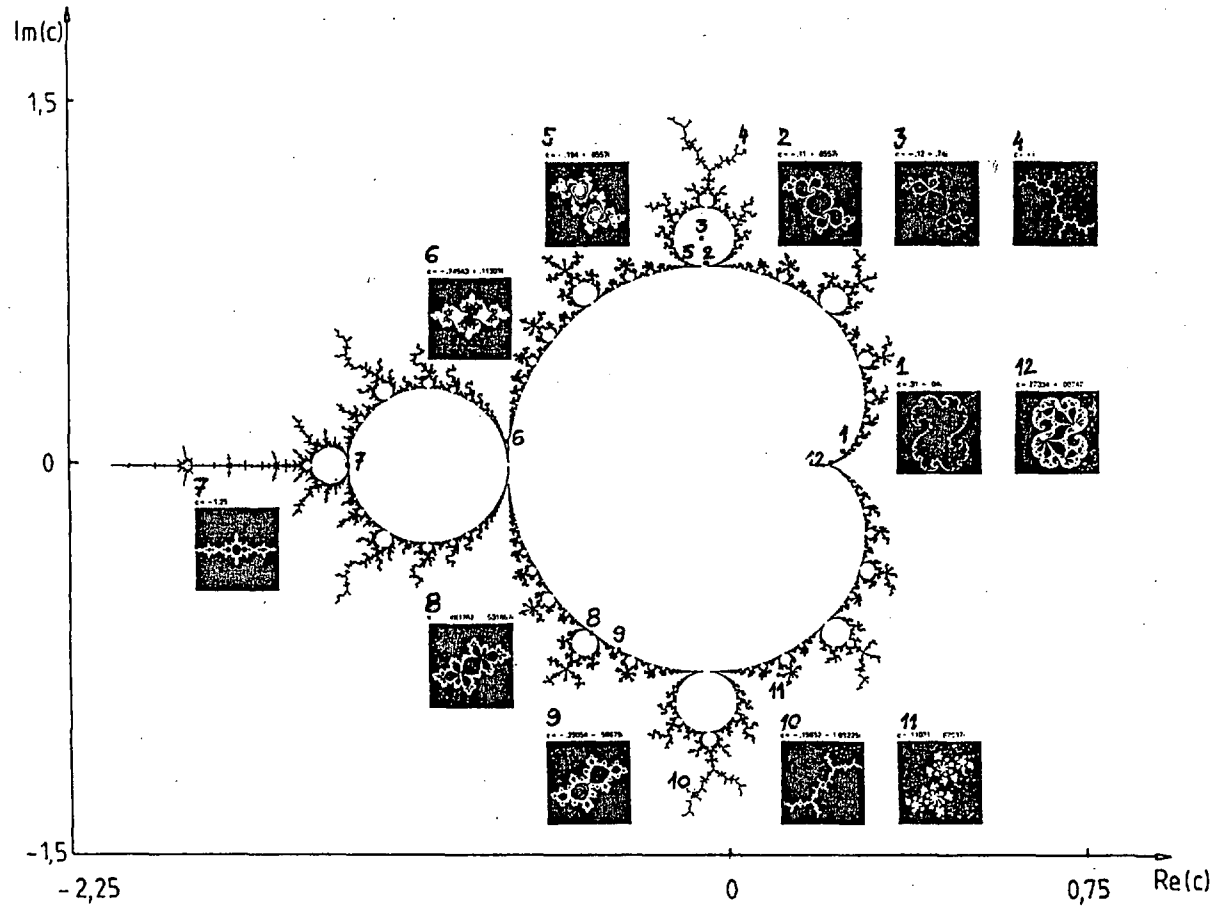
2.1 Podobnosti Mandelbrotove množice z Verhulstovim dinamičnim procesom

Raziskave so pokazale, da ima Mandelbrotova množica veliko skupnih točk z Verhulstovim dinamičnim procesom (slika 2.14). Pri Mandelbrotovi množici ugotovimo stabilno področje ter oscilacije s periodo $2, 4, 8, 16, \ldots$, ravno tako kakor pri Verhulstovem dinamičnem procesu.

Stabilno področje zajema območje med $c=0.25$ in $c=-0.75$, oscilacije s periodo 2 se pojavljajo v krogu s polmerom 1/4 in središčem v točki $c=-1$, oscilacije s periodo 4 pa okoli središča $c=-1.3107$ itn. Točke, kjer nastajajo "popki" na realni osi na sliki Mandelbrotove množice, ustrezajo točkam, kjer prihaja do podvajanj periode v Verhulstovem procesu.



Slika 2.14 Podobnosti Mandelbrotove množice z Verhulstovim dinamičnim procesom

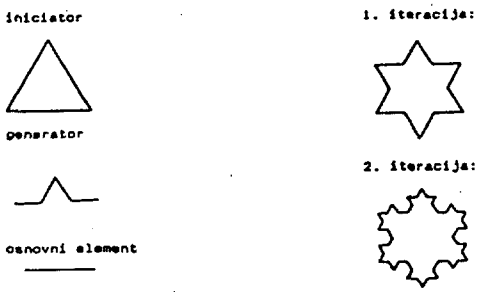


Slika 2.13 Juliove množice kot sestavni del Mandelbrotove množice

3. GEOMETRIČNI FRAKTALI

Geometrični fraktali se od naravnih razlikujejo po tem, da ustvarjajo slike, ki jih je mogoče vnaprej predvideti. Primer geometričnega fraktala je Kochova snežinka (slika 3.1), ki jo lahko generiramo s pomočjo naslednjega algoritma:

```
Kochova_krivulja := iniciator;
loop
    vsak osnovi element v Kochovi_krivulji
    zamenjaj z generatorjem
endloop.
```

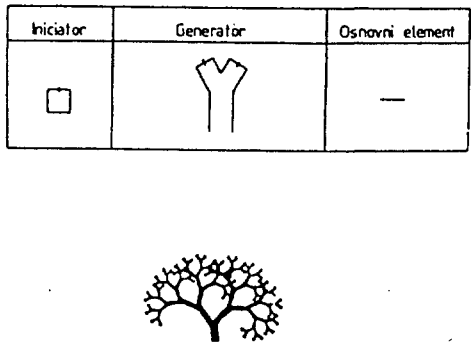


Slika 3.1 Primer geometričnega fraktala-Kochova snežinka

Opazimo lahko, da s ponavljanjem iteracij v globino ustvarjamo čedalje bolj natančno sliko. S tem dobimo občutek, da je objekt (snežinka), ki ga opazujemo, zelo blizu. Pri majhnem številu iteracij dobimo občutek oddaljenosti objekta.

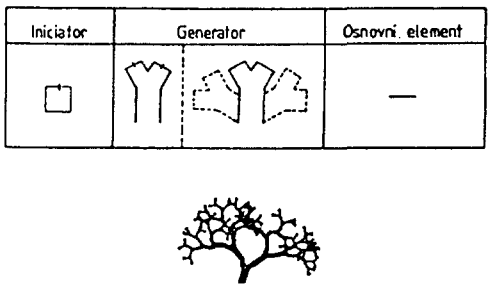
Kochova snežinka predstavlja najprimitivnejši fraktal v verigi geometričnih fraktalov. To pomeni, da lahko v osnovni algoritem uvedemo dodatna pravila in na tak način dobimo zanimivejše in bolj kompliciran fraktale.

Pravilo 1: Nekatere daljice iniciatorja zaznamujemo kot koristne in jih v iteracijskem postopku spreminjamo (slika 3.2).



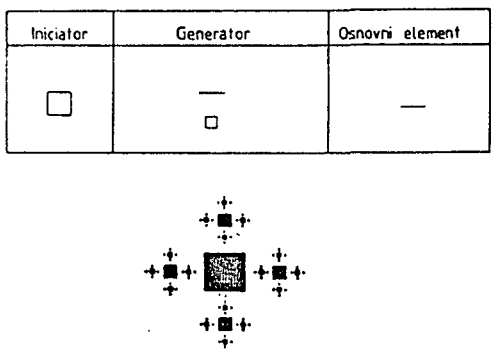
Slika 3.2 Drevo, narisano s pomočjo pravila 1 in zapolnitvijo notranjosti

Pravilo 2: Dodamo funkcijo, ki bo dovoljevala generatorju naključni premik na levo ali desno (slika 3.3).



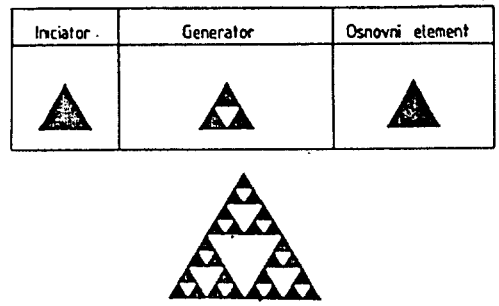
Slika 3.3 Drevo, narisano s pomočjo pravila 2 in zapolnitvijo notranjosti

Pravilo 3: Generator sestavimo iz dveh ali večih delov (slika 3.4).



Slika 3.4 Vzorec, narisan s pravilom 3 in zapolnitvijo kvadratkov

Pravilo 4: Predpostavimo, da je osnovni element enak iniciatorju (slika 3.5).



Slika 3.5 Trikotnik Sierpinskega, narisan s pomočjo pravila 4

Seveda lahko ta pravila združimo in jih uporabimo skupaj. Na tak način porušimo geometrijsko natančnost in dobimo nepravilne krivulje, ki imajo obliko oblakov, gorovja itn.

4. UPORABA METODE IFS ZA GENERIRANJE GEOMETRIČNIH FRAKTALOV

Trikotnik Sierpinskega iz pravila 4 je mogoče narisati tudi na na drugačen način in sicer tako, da izberemo začetno točko $\vec{p}_{0,h} = [0,0,1]$ (dano s homogenimi koordinatami) ter eno izmed treh transformacij:

$$T_1 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ T_1 je skalirna matrika.

$$T_2 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$ T_2 je produkt skalirne in translacijske matrike.

$$T_3 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$ T_3 je produkt skalirne in translacijske matrike.

Transformacijo izberemo naključno, jo izvršimo nad začetno točko in dobimo točko $\vec{p}_{1,h}$. Nato naključno izberemo naslednjo transformacijo in jo izvršimo nad točko, dobljeno v prejšnjem koraku. Verjetnosti za izbiro posamične transformacije so enake. Postopek nadaljujemo in po določenem številu iteracij pridemo do trikotnika Sierpinskega.

Na prvi pogled se zdi čudno kako lahko naključna izbira ene od treh transformacij privede v iterativnem postopku do tako složne harmonije, kakor je trikotnik Sierpinskega. Področje v matematiki, ki se ukvarja s takšnimi transformacijami, je teorija iterativnih funkcijskih sistemov (IFS) in od tod tudi ime za našo metodo.

5. UPORABA FORMALNIH JEZIKOV NA PODROČJU FRAKTALNE TEORIJE

Nekatere objekte, na primer drevje in rastlinje, je mogoče predstaviti s pomočjo gramatik PRG (parallel rewriting grammars). Benoit Mandelbrot v svojih definicijah [6] ne dovoljuje , da bi objekte, dobljene na tak način, imenovali fraktali. Trdi namreč, da je pojem fraktala strogo geometrijski, medtem ko s pomočjo formalnih jezikov dobimo strukturirano zgradbo.

Objekte, nastale s pomočjo gramatik, imenujemo *graftali* [8]. Graftali se od fraktalov razlikujejo po tem, da ne ustvarjajo strogo zrcalnih in samopodobnih slik ter ne dovoljujejo nikakršne naključnosti pri uporabi produkcijskih pravil.

Značilnost gramatik PRG je zaporedna uporaba produkcijskih pravil (nad vsakim simbolom izvršimo produkcijsko pravilo), prav tako ne ločijo terminalnih simbolov od neterminalnih.

Oglejmo si primer!

Dana naj bo abeceda $\Sigma = \{0, 1, (,), [,]\}$

začetni simbol S=0 ter

produkcijska pravila:
$0 \rightarrow 1[0]1(0)0$
$1 \rightarrow 11$
$[\rightarrow [$
$] \rightarrow]$
$) \rightarrow)$
$(\rightarrow ($

Uporabimo naslednji algoritem:

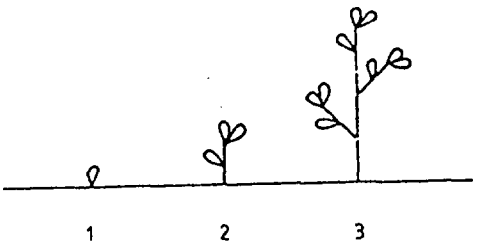
```
for i:= 1 to stevilo_generacij do
  nad vsakim simbolom v nizu uporabi produkcijsko
  pravilo
```

Pri stevilu_generacij = 3 dobimo:

stevilo_generacij	niz
1	0
2	1[0]1(0)0
3	11[1[0]1(0)0]11(1[0]1(0)0)1[0]1(0)0

Dobljeni niz želimo grafično predstaviti v obliki drevesa, zato uporabimo preprosto interpretacijo:

0...nariši list
1...nariši črto
[...začetek vejitve v levo
]...konec vejitve v levo
(...začetek vejitve v desno
)...konec vejitve v desno



Slika 5.1 Prikaz postopka, kako s pomočjo predhodno definirane gramatike tvorimo graftalno sliko. Vejitveni kot znaša 45 stopinj.

Seveda obstajajo še bolj kompliciranе interpretacije našega niza. Tako lahko npr. spreminjamo vejitveni kot glede na smer vetra, čvrstost, velikost drevesa itn. Prav tako lahko predpostavimo veje v obliki valja, katerega premer in višina sta odvisna od položaja enice v nizu. Ta dodatni pogoj omogoča, da v določenem trenutku dobimo manjše in tanjše veje.

6. OPIS ALGORITHOV ZA GENERACIJO FRAKTALOV

Pri oblikovanju 2D fraktalnih slik smo upoštevali dinamični proces nad polinomom druge stopnje $z \Rightarrow z^2 + c$ (namesto x pišimo z). Kompleksni števili z in c smo razdelili na realni in imaginarni del (z=x+iy, c=p+iq) tako, da je dinamični zakon prešel v obliko:

$$x_{n+1} = f(x_n, y_n, p) = x_n^2 - y_n^2 + p$$
$$y_{n+1} = g(x_n, y_n, q) = 2x_n\, y_n + q$$

Kot že omenjeno, smo do oblikovanja slik prišli po dveh poteh:

1. Upoštevali smo ravnino xy in fiksni vrednosti p ter q. Za vsako točko ravnine smo ugotovili njeno dinamično povezanost do ustreznega stekališča. Na tak način smo raziskali strukturo Juliovih množic.
2. Upoštevali smo ravnino pq ter fiksno točko (x,y). Tak postopek je produciral slike Mandelbrotove množice.

Program za izračun dinamičnega procesa ter določitev barv je tekel na VAX 8800. Rezultate je shranjeval na datoteke, ki smo jih kasneje prenesli na PC-AT. Tam je tekel program za izris fraktalnih slik. Napisan je bil v turbo pascalu.

Pri izrisu slik smo uporabili 16 barv. Povedale so, kako dolgo je potrebovala točka (x,y) da se je približala stekališču. Če se točka po določenem številu iteracij (5000) ni približala stekališču, smo jo pobarvali s črno barvo. Velikost okna (a x b) smo izbrali (511 x 350) pikslov.

Algoritma 1 in 2 smo uporabili za generiranje naključnih fraktalov. S pomočjo algoritma 3, pa smo

narisali geometrična fraktala trikotnik Sierpinskega in list praproti.

Algoritem 1: Juliova množica

korak 0: Izberi parameter c=p+iq
Določi velikost okna: $x_{min} = y_{min}$
$x_{max} = y_{max}$
$\Delta x=(x_{max} - x_{min})/(a-1)$
$\Delta y=(y_{max} - y_{min})/(b-1)$
Za vse točke zaslona (n_x,n_y), $n_x = 0,1,\ldots,a-1$, $n_y = 0,1,\ldots,b-1$, izvrši naslednje korake.

korak 1: $x_0 = x_{min} + n_x\Delta x$
$y_0 = y_{min} + n_y\Delta y$
števec_iteracij = 0

korak 2: $x_{n+1} = x_n^2 - y_n^2 + p$
$y_{n+1} = 2x_n y_n + q$
Povečaj števec_iteracij za 1.

korak 3: Izračunaj $r^2 = x_n^2 + y_n^2$
(i) if r>2 then izberi barvo in goto korak 4
(ii) if števec_iteracij = 5000 then izberi barvo 0 (črno) in goto korak 4
(iii) if r ≤ 2 and števec_iteracij<5000 then goto korak 2

korak 4: Pobarvaj točko (n_x,n_y) z ustrezno barvo, izberi naslednjo točko in pojdi na korak 1.

Da bi izboljšali čas računanja, smo upoštevali, da točki (x,y) in (-x,-y) producirata enak rezultat, saj je slika simetrična glede na izhodišče. Pozorni moramo biti tudi na izbiro koordinat okna, saj ob njihovi nepravilni izbiri lahko dobimo nezanimivo sliko.

Algoritem 2: Mandelbrotova množica

korak 0: Izberi p_{min}, p_{max}, q_{min}, q_{max}.
$\Delta p = (p_{max} - p_{min})/(a-1)$
$\Delta q = (q_{max} - q_{min})/(b-1)$
Za vse točke zaslona (n_p,n_q), $n_p = 0,1,\ldots,a-1$, $n_q = 0,1,\ldots,b-1$, izvrši naslednje korake.

korak 1: $p_0 = p_{min} + n_p\Delta p$
$q_0 = q_{min} + n_q\Delta q$
števec_iteracij = 0
$x_0 = y_0 = 0$

korak 2: $x_{n+1} = x_n^2 - y_n^2 + p$
$y_{n+1} = 2x_n y_n + q$
Povečaj števec_iteracij.

korak 3: Izračunaj $r^2 = x_n^2 + y_n^2$
(i) if r>2 then izberi barvo goto korak 4
(ii) if števec_iteracij=5000 then izberi barvo 0 (črno) in goto korak 4
(iii) if r ≤ 2 and števec_iteracij<5000 then goto korak 2

korak 4: Pobarvaj točko (n_p,n_q) z ustrezno barvo, izberi naslednjo točko in pojdi na korak 1.

Pri tretjem algoritmu smo začetno točko (x_0,y_0) izbrali (0,0). Število iteracij je bilo 32000. Program je v celoti tekel na PC.

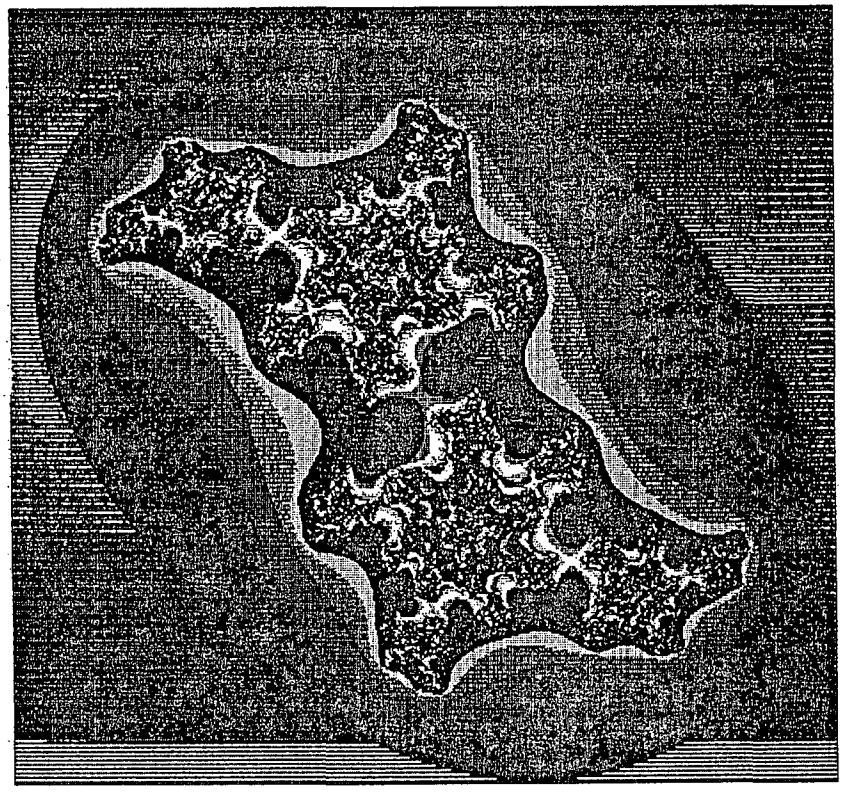
Algoritem 3 : metoda IFS

korak 0: Okno opazovanja V naj bo velikosti X × Y, pri čemer je X × Y resolucija zaslona.
Okno V razdelimo na L × M kvadratov (označimo jih z V_{ij}) velikosti dolž (L=L/dolž, M=M/dolž).
Izberi število_iteracij ≥ L × M.
Polje V inicializiraj na 0.
Izberi začetno točko $(x_0,y_0) \in R^2$.
Število_barv=16.

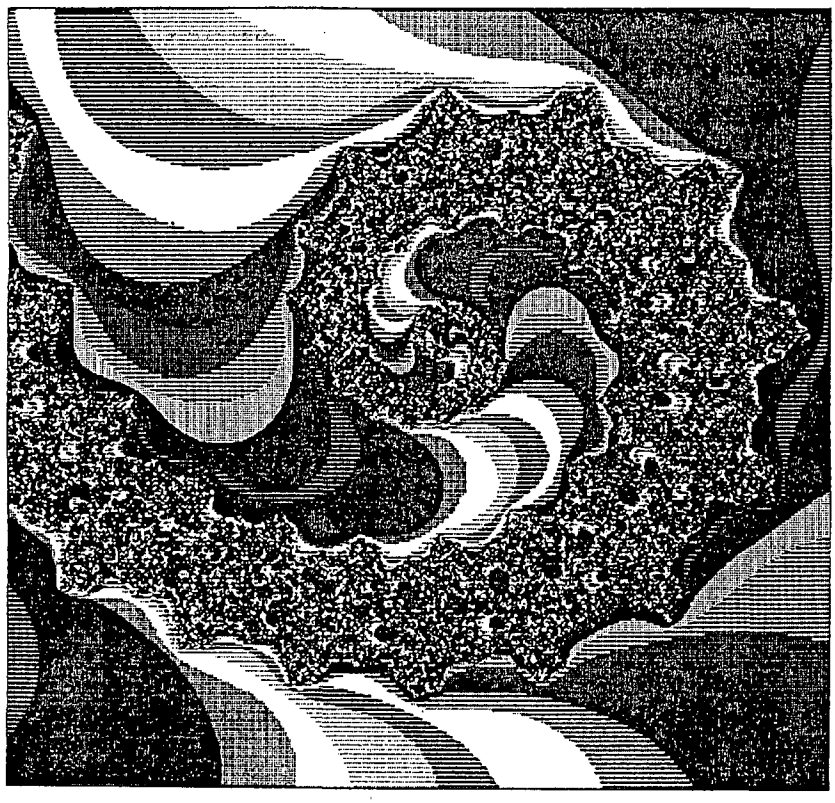
korak 1:
```
for n = 0 to število_iteracij do
begin
   (* izberi nakjučno transformacijo M_k *)
   rand = random število med [0,1];
   verjetnost = p_1;
   k = 1;
   while (verjetnost < rand) do
   begin
      k = k+1;
      verjetnost = verjetnost + p_k;
   end;
   (* izvrši transformacijo nad točko *)
   [ x_{n+1} y_{n+1} 1 ] = [ x_n y_n 1 ] M_k
   n_1 = int(x_{n+1});
   n_2 = int(y_{n+1});
   (* zaznaj "obisk" v kvadratu V_ij *)
   V[n_1][n_2] = V[n_1][n_2] + 1;
end;
```

korak 2: Poišči maksimalno vrednost max v polju V.
Določi barvo pravokotnika V_{ij}:
barva_pravokotnika=barva(V[i][j]*15/max)
in ga pobarvaj.

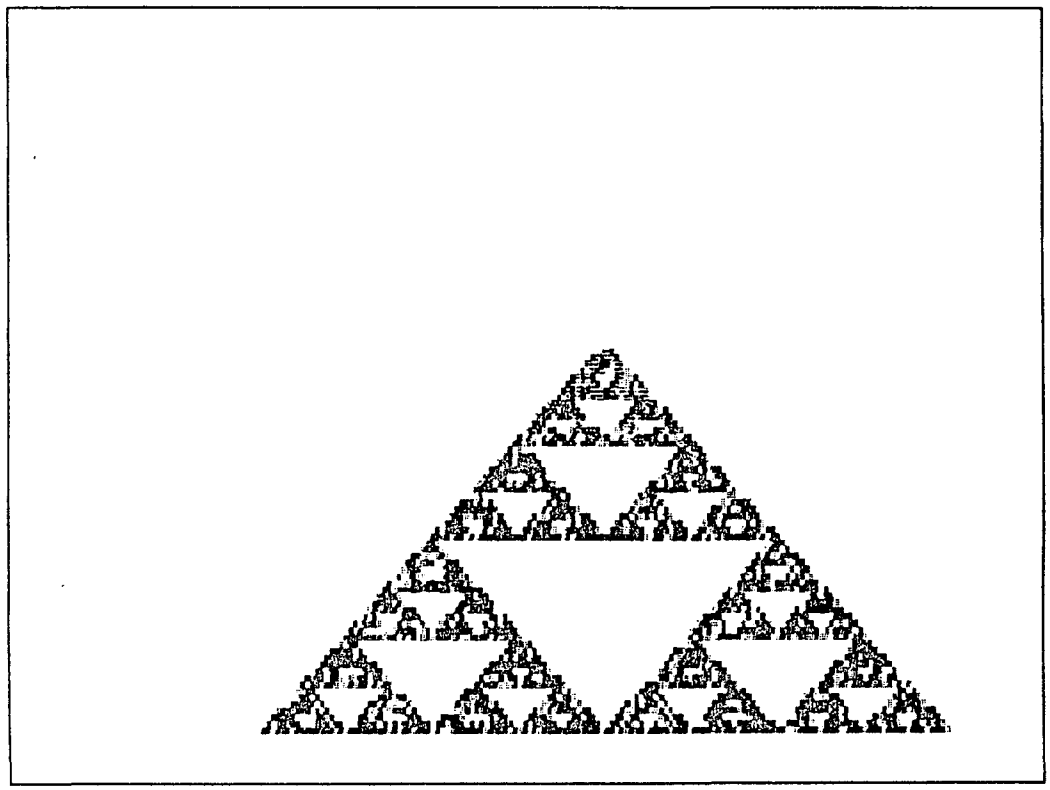
Slike 6.1, 6.2 in 6.3 so bile izrisane na barvnem brizgalnem risalniku Tektronix 4696.



Slika 6.1 Juliova množica pri c = 0.11031 - 0.670371



Slika 6.2 Mandelbrotova množica



Slika 6.3 Trikotnik Sierpinskega dobljen s pomočjo metode IFS

7. ZAKLJUČEK

Znanost in umetnost sta dva nasprotujoča si načina za izražanje naravnega sveta - prvi analitičen, drug intuitativen. S pomočjo fraktalov smo ti dve poti združili in dokazali, da sta odvisni druga od druge.

Fraktali so zanimivi zaradi visoke stopnje vizualne kompleksnosti, čeprav so v svojem bistvu izredno preprosti. Matematična osnova fraktalne teorije je enostavna, saj temelji na ponavljanju. Prav zaradi tega jo je bilo izredno lahko uporabiti na področju računalništva. Algoritmi za kreiranje fraktalnih slik potrebujejo malo podatkov, s pomočjo rekurzije pa so zelo uspešni.

Teorija fraktalov je pomagala zlomiti strogo geometrijo likov. Trendi v svetu pa segajo še dalje. Področje uporabe fraktalov se naglo širi - od biologije do tekstilne industrije pa vse do drugih, predvsem barvnih grafičnih izdelkov. Možnosti je torej ogromno - ta članek opisuje le delček njih.

8. LITERATURA

[1] M.Barnsley: "Harnessing Chaos for Image Synthesis", Computer Graphics, Volume 22, Number 4, avgust 1988

[2] P.Blanchard: "Complex Analytic Dynamics on the Riemann Sphere", Amer.Math Soc. 11, 1984, str.85-141

[3] S.Demko: "Construction of Fractal Objects with Iterated Function Systems", Computer Graphics, Volume 19, Number 3, 1985, str.271-278

[4] M.Gervautz: "Fractals in Computer Graphics", Proceeding of The Second Austro-Yugoslav Conference on Computer Graphics, str.96-103

[5] J.A.Kaandorp : "Interactive Generation of Fractal Objects", Eurographics'87, str.181-196

[6] B.B.Mandelbrot: "The Fractal Geometry of Nature", W.H.Freeman, San Francisco, 1982

[7] H.O.Peitgen, P.H.Richter: "The Beauty of Fractals", Springer Verlag, Berlin, Heidelberg, New York, 1986

[8] A.R.Smith: "Plants, Fractals and Formal Languages", Computer Graphics, Volume 18, Number 3, julij 1984, str.1-10

[9] A.Terčelj: "Fraktali - grafične skrivnosti računalniških umetnikov", Informatica 3/87, str.46-50

PRIMENA METODA INŽENJERSTVA ZNANJA U OBRAZOVANJU

INFORMATICA 4/89

Keywords: method, knowledge engineering, artificial intelligence, education

Ljubomir Jerinić, Zoran Budimac, Đura Paunić i Mirjana Ivanović

U radu su opisane metode inženjerstva znanja kao podoblasti veštačke inteligencije, i mogućnosti primene tih metoda u obrazovanju. Kao najpogodnija metoda reprezentacije znanja za potrebe obrazovanja izabran je sistem okvira Marvina Minskog. Na osnovu takvog pristupa razvijen je programski sistem OSOF za prikupljanje, internu reprezentaciju i korišćenje znanja u svim vidovima i nivoima obrazovanja.

1. UVOD

Prema [1], veštačka inteligencija (VI) je deo računarskih nauka u kojima se projektuju inteligentni računarski sistemi, koji se ponašaju na način sličan inteligenciji u ljudskom ponašanju, za razumevanje prirodnih jezika, učenje, rezonovanje, rešavanje problema i sl. VI je interdisciplinarna nauka i potekla je iz istraživanja iz domena simboličke obrade podataka i dela psihologije o rasuđivanju. VI istražuje simboličku obradu i heurističke procese zaključivanja i rezonovanja, kao i predstavljanje znanja u obliku pogodnom za zaključivanje uz pomoć računara [2].

Osnovni pravci istraživanja u okviru VI su: razumevanje prirodnog govora, mašinsko učenje, automatsko programiranje, računarska vizija, inteligentna robotika, inženjerstvo znanja itd.

Razvojem računarske tehnologije i metoda VI u zadnjih deset godina, približavanjem računara svim uzrastima, kao i uvođenjem računara u škole, istraživačima VI obrazovanje postaje interesantno polje rada. Deo VI, inženjerstvo znanja, po svojoj definiciji, metodama i rezultatima postaje direktno primenljivo i u obrazovanju, sa ciljem krajnje individualizacije obrazovnog procesa. Davnašnja težnja da jedan nastavnik ili profesor obrazuje jednog učenika ovakvim pristupom postaje realnost, kao i napredovanje svakog učenika prema njegovim sposobnostima.

U radu se dalje opisuju metode inženjerstva znanja i njihova primena u obrazovanju. Data je definicija i klasifikacija načina pretstavljanja znanja, prihvaćena u inženjerstvu znanja. Iskorišćena je metoda M. Minskog [6] za realizaciju univerzalnog programskog paketa OSOF [3] za primenu računara u obrazovanju.

2. INŽENJERSTVO ZNANJA

U ranom razdoblju istraživanja u ovoj podoblasti VI, do sredine 70-tih godina, težilo se (pod uticajem psihologije), iznalaženju opštih metoda rešavanja problema "ekspertize" znanja, zasnovanih na opštim principima zaključivanja sa psihološkog aspekta. Ovakav pristup se pokazao neefikasnim za Von Neuman-sku organizaciju računara i sa malom primenljivošću u praksi. Nedostatak je prevashodno što se unutar opšteg generalizovalo specifično znanje relativno disjunktnih oblasti.

Krajem sedamdesetih godina se sa paradigme zasnovane na zaključivanju prešlo na novu paradigmu zasnovanu na znanju. Oblast delovanja istraživača inženjerstva znanja postaje istovetna sa oblastima delovanja stručnjaka iz pojedinih uskih oblasti: prikupljanje specifičnih znanja i iskustava, te potom i njihova primena u rešavanju određene grupe problema. Preduslov za ovakvo, heurističko, rešavanje problema je izbor pogodne reprezentacije relevantnog znanja kome inteligentni program može lako da pristupi, dok mehanizam zaključivanja, tj. mehanizam korišćenja tako memorisanog znanja treba da je jednostavan i zasnovan na tom znanju umesto na opštim principima ili nekakvim funkcijama komplikovanim za izračunavanje.

Ljudsko znanje se kodira određenim metodama u module koji se u mehanizmu zaključivanja aktiviraju uzorcima: "sirovi" podaci, "obrađeni" podaci, parcijalna rešenja, neočekivane situacije, greške i sl. Ovakvi uzoračko vođeni moduli imaju niz prednosti u odnosu na nekakav opšti algoritam zaključivanja:

-predstavljanje znanja u delovima je primereniji načinu memorisanja znanja eksperata,

-programiranje sa ovakvim modulima omogućava razvoje inteligentnih sistema u koracima, programi se lako modifikuju i proširuju, a greške unutar znanja se popravljaju bez izmene koda programa, i sl.

Proces konstruisanja jednog inteligentnog sistema obuhvata sledećih pet oblasti:

-prikupljanje i sistematizacija relevantnog znanja: od eksperta ili mašinskim učenjem na primerima,

-reprezentacija znanja: izbor pogodnog načina kojim se velika količina znanja može predstaviti pomoću simboličkih struktura podataka unutar računara, pogodnih za zaključivanje. Odabrana struktura treba da omogući fleksibilne izmene i dopune memorisanog znanja,

-primena znanja: planiranje i kontrola rešavanja problema, heurističko zaključivanje, tačnost i efikasnost rada inteligentnih procesa, koje memorisano znanje predstavlja,

-generisanje objašnjenja: interakcija računar-čovek, objašnjenje zašto je problem rešen na jedan a ne na neki drugi način, mogućnost učenja na greškama i uticanja čoveka na proces zaključivanja,

-obrazovanje: smešteno znanje se uz izvesne ograde može koristiti i za stvaranje

inteligentnih sistema učenja.

3. INŽENJERSTVO ZNANJA I OBRAZOVANJE

Računar u obrazovanju sa stanovišta nastave se može posmatrati kao novo nastavno sredstvo i kao upravljač nastavnog procesa. Za razliku od konvencionalnih nastavnih sredstava (grafoskop, diaskop, interna televizija, responderi i dr.) računar u nastavni proces unosi značajnu novinu - mogućnost obostrane komunikacije računar - učenik. Nijedno od konvencionalnih nastavnih sredstava ne može da odgovara na pitanja učenika i da na taj način usmerava nastavni proces. Kao upravljač nastavnog procesa, računar u obrazovanju se posmatra kao neposredni izvršioc nastavnog procesa kreiranog putem programske podrške i upravljač raznih pomoćnih uređaja nastavnog procesa (laboratorijski uređaji, responderi i dr.).

Ovako posmatrana primena računara u obrazovnom procesu otvara mogućnost primena metoda i tehnika inženjerstva znanja u obrazovanju i otvara novo polje istraživanja - projektovanje inteligentnih sistema učenja. Projektovanje ovakvih sistema učenja zahteva interakciju metoda inženjerstva znanja sa jedne strane i metoda pedagogije, metodike, didaktike i psihologije sa druge.

Pretpostavka da se znanje koje sadrže ekspertni sistemi standardnog tipa može iskoristiti i u svrhe obučavanja, demantuje se u praksi jer su takvi inteligentni programi loši učitelji [4]. Uzrok je upravo izostavljanje osnovnih principa pedagogije i metodike pri kreiranju programskih sistema čija je svrha prvenstveno konsultanska pomoć pri rešavanju problema.

Kreiranje posebnih inteligentnih sistema namenjenih obrazovanju, mora rešiti sledeće probleme:

-kako predstaviti znanje koje učenik treba da usvoji,

-kako opisati pojmove koji se usvajaju,

-kako usaglasiti mehanizam korišćenja takvog znanja sa metodičkim i pedagoškim principima usvajanja znanja,

-kako iskoristiti dobre osobine svih vrsta konvencionalnih načina prenošenja i usvajanja znanja: predavanje, testiranje, eksperiment, opažanje, programirana nastava i dr.

4. SISTEMI REPREZENTOVANJA ZNANJA

Znanje se može definisati kao simbolička reprezentacija činjenica i relacija među njima. Reprezentacija znanja je način predstavljanja znanja, kao i način kako se ono može povezivati sa drugim znanjem, koristiti za rešavanje novih situacija i izvoditi novo znanje.

U [8] reprezentacija znanja se definiše kao skup sintaksnih i semantičkih konvencija koje omogućavaju opisivanje stvari. Sintaksa reprezentacije je skup pravila za formiranje, kombinovanje i uređenje izraza u jeziku reprezentacije. Semantika reprezentacije određuje kako se sintaksno valjani izraz koji reprezentuje znanje interpretira, tj. kako se iz date forme može izvesti i koristiti znanje.

Reprezentacija znanja je bitan element inženjerstva znanja koji omogućava sistematičan način kodiranja znanja stručnjaka o nekom problemu unutar neke šire kategorije. Ona implicitno obuhvata i organizaciju podataka u računaru za memorisanje znanja. Postoji niz notacijskih sistema za predstavljanje znanja u računaru. Zajednička karakteristika svih je da treba da obezbede brz pristup znanju i da se znanje može lako koristiti pomoću manje-više prirodnih mehanizama zaključivanja ili izvođenja. Programeru nakon odabiranja notacijskog sistema ostaje da izabere najpogodniju strukturu podataka koja se implementira zadržavajući sve dobre osobine izabranog notacijskog sistema.

Važni kriterijumi za izbor notacijskog sistema i organizaciju podataka za reprezentaciju znanja su:

-logička dovoljnost tj. da formalizam može na pogodan način da opiše znanje,

-efikasnost pristupa pri obradi znanja,

-iskoristivost, tj. reprezentovano znanje se može upotrebiti u rešavanju problema za koje su ta znanja dovoljna.

Za reprezentaciju znanja u primeni inženjerstva znanja se koriste sledeći formalizmi: produkcijska pravila, strukturni objekti i predikatska logika.

Svi navedeni formalizmi se implementiraju sami ili u nekoj kombinaciji, u uzoračko vođenim sistemima. Struktura podataka koja pretstavlja unutrašnju reprezentaciju izabranih formalizama pretstavljanja znanja je zavisna od problema koji se rešava, sistema zaključivanja, interpretatora mehanizma procesiranja znanja, računara na kome se implementira i programskog jezika izabranog za realizaciju određenog inteligentnog sistema.

Sistemi koji koriste neki od navedenih formalizama, se sastoje od niza relativno nezavisnih modula koji omogućavaju prikupljanje i memorisanje znanja, formiranje odgovarajuće unutrašnje reprezentacije znanja, korišćenje prikupljenog znanja i sl.

Produkcijska pravila su formalizam za predstavljanje znanja koji se koristi i u teoriji automata, formalnim jezicima i dizajniranju programskih jezika. Sastoje se od skupa pravila oblika:

$$\text{if } (P_1 \wedge \ldots \wedge P_n) \text{ then } (A_1 \wedge \ldots \wedge A_m),$$

gde su P_i, i=1,...,n uslovi, a A_j, j=1,...,m zaključci.Uslovi su trojke objekt - atribut - vrednost oblika:

(Bor je_rudnik bakar)

dok su zaključci akcije koje se preduzimaju ako su uslovi zadovoljeni.

Korišćenjem ovog formalizma kodiraju se iskustvene veze (asocijacije) između uzoraka podataka i akcija koje sistem treba da preduzme kao posledicu zadovoljenosti uslova. Produkcijskim pravilima se kodiraju dva tipa znanja: opšte znanje relevantno za problem koji se opisuje tim znanjem i specifično znanje važno za posebne primene koje će koristiti to znanje. To specifično znanje se može i reprezentovati tradicionalnim tehnikama pretstavljanja znanja u bazama podataka. Neodređeno znanje koje, se takođe može koristiti je pretstavljeno trojkom:

$$\text{if } (P_1 \wedge \ldots \wedge P_n) \text{ then}$$

$$(A_1 \wedge \ldots \wedge A_m) \text{ with } V(0.7)$$

gde je V(0.7) verovatnoća da $(P_1 \wedge \ldots \wedge P_n)$ izvodi $(A_1 \wedge \ldots \wedge A_m)$.

Predikatska logika, u čijoj osnovi je propozicioni i predikatski račun, razvojem logičkog programiranja i logičkih programskih jezika, se koristi i kao sistem predstavljanja znanja. Znanje se predstavlja pomoću formula, kao npr.:

Svako voli nekog <==> $(\forall x)\ (\exists y)$ (voli(x y))

Ovakav načina opisivanja znanja se koristi u

reprezentaciji opštih činjenica za razliku od produkcijskih pravila u kojima znanje predstavlja iskustvene činjenice.

Strukturni objekti su opšti izraz za bilo koju šemu reprezentacije znanja. Osnovni delovi strukturnih objekata su analogni čvorovima i vezama u teoriji grafova, otvorima ("slot") i markerima teorije okvira ("frame"), ili otvorima i puniocima ("filler") konvencionalnih struktura podataka tipa slog. Oni obuhvataju sledeće formalizme: semantičke (asocijativne mreže), okvire, objektno orjentisane sisteme, strukture konceptualne zavisnosti ili skripte.

Predikatska logika i produkcijski sistemi kao formalizmi predstavljanja znanja se koriste za reprezentovanje različitih aspekata okoline. Međutim, u opštem slučaju, oni ne dozvoljavaju da struktuiramo znanje o toj okolini. Strukturni objekti uključuju mogućnost reprezentovanja strukture znanja, grupisanjem delova znanja u celine i relacija među znanjem, kao i pokazivača na akciju koja se preduzima korišćenjem tog znanja. Takođe se unutar strukturnih objekata predviđa rad sa podrazumevanim (default) vrednostima, otkrivanje grešaka kao i rad sa nepotpunim znanjem. Sve ove nabrojane osobine su direktno primenljive u predstavljanju znanja za obrazovne svrhe.

5. PREDSTAVLJANJE ZNANJA U SISTEMU OSOF

Znanje za potrebe obrazovanja se može organizovati u nastavne sekvence (lekcije) koje se sastoje od skupa međusobno povezanih članaka (pojmova). Članak sadrži minimalnu količinu zaokruženog znanja koga ucenik treba da savlada u jedinici vremena.

Svaki članak se sastoji od naziva, teksta kojim je on opisan, grafičke ilustracije ili simulacije i proizvoljnog broja zadataka kojim se može proveriti i utvrditi znanje opisano u članku.

Zadatak je opisan tekstom, slikom ili simulacijom, a odgovor se može eksplicitno uneti, odabrati od niza ponuđenih alternativa ili odabrati iz dve grupe ponuđenih alternativa, uparivanjem.

Alternative su opisane tekstom, a svaki mogući odgovor sadrži i informaciju o daljem toku učenja u slučaju da je učenik odgovorio na odgovarajući način ili odabrao određenu alternativu kao svoj odgovor na postavljeni zadatak.

U sistemu OSOF iskorišćena je metoda okvira za pretstavljanje ovako organizovanog znanja. Okvir se sastoji od skupa imenovanih slotova koji sadrže vrednosti ili pokazivače na druge okvire [5]. Struktura podataka koja odgovara navedenoj reprezentaciji i organizaciji znanja zapisana u pseudo jeziku za opis okvira je:

```
Begin Sekvenca
      ( Ime: Niska;
       F Početak: Članak )
Begin Članak
      ( Pojam:    Niska;
        Opis:     Niska, Simulacija, Slika;
       F Pitanje: Zadatak  )
End;
Begin Zadatak
      ( Opis:     Niska, Simulacija, Slika;
       F Sledeći: Zadatak;
       F Odgovor: Alternativa )
End;
Begin Alternativa
      ( Opis:     ( Izbor, Otvoreni odgovor,
                    Uparivanje );
       F Akcija:  ( Članak, Zadatak,
                    Dopuna_opisa, Kraj );
       F Sledeći: Alternativa )
End;
Begin Kraj
      ( Ime: Niska;
       F Akcija: ( Sekvenca, Članak, 'Exit' ) )
End;
```

U okvirima Sekvenca, Članak, Zadatak i Kraj "Niska" označava tekst proizvoljne dužine, "Simulacija" proceduru za simuliranje nekog procesa, a "Slika" proceduru za grafički prikaz proizvoljne slike. Unutar okvira Alternativa mogući zadaci koje učenik treba da izvrši su: "Izbor" - na postavljeni zadatak se odgovara izbor jedne od navedenih alternativa, "Otvoreni odgovor" - zahtev za eksplicitni upis rešenja i "Uparivanje odgovora" - postavljeni zadatak se rešava uparivanjem odgovarajućih alternnativa iz dva skupa. Na osnovu učenikovog rešenja zadatka, akcija računara se odvija prelaskom na: jedan od okvira Članak, Zadatak i Kraj, ili grupu okvira Članak koji pružaju dodatno objašnjenje nejasnog opisa ("Dopuna opisa"). Oznaka F ispred imena slota označava da je njegova vrednost pokazivač na okvir.

6. SISTEM OSOF

Sistem OSOF [3] je razvijen u Institutu za matematiku u Novom Sadu i sastoji se iz modula za prikupljanje znanja TEA, njegovu internu reprezentaciju po opisanoj šemi i dva modula elementarnog korišćenja za usvajanje znanja: izborom alternativa - LEA i testiranjem - EXA. Realizovan je programom vođenim menijima i izuzetno je jednostavan za korišćenje, te od nastavnika i učenika ne zahteva nikakvo znanje o ustrojstvu računara niti znanje programiranja. Primenom sistema zaključeno je da je odabrana metoda okvira pogodna za primenu u obrazovanju.

LITERATURA:

1. Barr A., Feigenbaum E. A., Handbook of Artificial Intelligence, Stanford University Computer Science Dept, Stanford, (1980), USA

2. Buchanan B. G., Feigenbaum E. A., Dendral and Meta-Dendral: their applications dimension, Artificial Intelligence, 11(1,2), 5-24, (1978), USA.

3. Paunić Đ., Jerinić Lj., Budimac Z., Ivanović M., Univerzalni programski paket za primenu računara u nastavi, u štampi.

4. Clancey W. J., Methodology for building an intelligent tutoring system, from "Methods and tactics in Cognitive Science", Lawrence Erlbaum Publishers, (1983), USA.

5. Frost R., Introduction to Knowledge Based Systems, Collins, London, (1986), Great Britain.

6. Ivanović M., Jerinić Lj., Paunić Đ., Budimac Z., On knowledge representation in education, Review of research, Institute of Mathematics, Novi Sad, (1988), (u štampi).

7. Nilsson N. J., Principles of Artificial Intelligence, Tioga Publishing, Palo Alto, (1980), USA.

8. Winston P., Artificial Intelligence, Addison - Wesley, Reading, (19770, USA.

strategija računalništva

strategy of computing

Ali IBM lahko postane največje telefonsko podjetje na svetu?

IBMova telekomunikacijska divizija pridela že 10% IBMovega dohodka. IBMov tekmec, podjetje AT&T, zasleduje IBM z zrcalno strategijo, ko se pomika v računalništvo iz svoje telekomunaikacijske baze. IBM ni nakupno osamil podjetja Rolm zaradi opustitve svojih želja, da prodre v telekomunikacijsko področje, temveč zaradi dozorelosti lastnih zamisli, kaj pomeni konvergenca med računalništvom in telekomunikacijami. Približevanje med računalništvom in telekomunikacijami postaja namreč za IBM realnost. Zato IBM ni opustil načrta, da postane največje telefonsko podjetje na svetu.

IBMov neposreden nakup podjetja Rolm, proizvajalca zasebnih telefonskih central, za znesek $1,25bn, in manjši deleži doseženi v podjetjih, kot je AT&Tjev konkurent MCI, so elemeti "velike telekomunakacijske strategije". Ta strategija je bila sprejeta na široko in njeno sporočilo je, da postaja močno poudarjano približevanje med računalništvom in telekomunikacijami povsem realno. Tudi samo podjetje AT&T, ki ostaja edini bistveni konkurent IBMa v svetu informacijske tehnologije, sledi IBMu z zrcalno strategijo, po poti pomika iz telekomunikacij v računalništvo.

Le pred nekaj leti sta se obe podjetji umaknili z opečenimi prsti. Računajó, da je AT&T zgubil $3bn z namero, da postane računalniško podjetje. Medtem pa je IBM umaknil svoj delež v MCI in v zadnjem letu prodal podjetje Rolm nemškemu Siemensu, ko so se izgube pričele kopičiti. Ta IBMova poteza je marsikoga presenetila. Podjetje IBM je namreč dokaj vztrajno pri gradnji tržišč in prodaja podjetja Rolm je bila za IBM nekaj neobičajnega. Seveda si je IBM zelo prizadeval potegniti podjetje Rolm v dobičkonosne posle. Mlado podjetje na zapadni obali pa se je vselej ostro soočalo z IBMovo umirjeno modro kulturo in tako mu je bilo v začetku dovoljeno, da obdrži svojo neodvisnost. Toda zaradi naraščajočih težavav je IBMova finančna divizija v letu 1987 omejila separatno strukturo podjetja Rolm. V letu 1988 so izgube narasle na $200m in IBM je moral ukrepati.

Po odprodaji podjetja Rolm so nekateri sklepali, da z zbliževanjem računalništva in telekomunikacij ne bo nič. Toda že brez podjetja Rolm je IBM pri svojih več kot $50bn letnega priliva pridelal 10% tega v svoji komunikacijski diviziji. Zanimivo je, da to divizijo vodi ženska (Ellen Hancock), ki sodi med najdaljnovidnejše IBMove menagerje. IBMova mrežna tehnologija za lokalne mrežne sisteme s t.i. token ringom je že danes vodilni mednarodni standard; sistemska mrežna arhitektura (SNA), ki je bila uvedena že l. 1974, je instalirana na več kot 30000 mestih po svetu. Medtem ko napovedi kažejo, da se bo npr. v Veliki Britaniji povečala uporaba OSI (Open System Interconnection) iz 3% v letu 1987 na 16% v letu 1992, bo to povečanje za SNA napredovalo iz 34% na 45%.

IBMova moč se nahaja v njegovi zapleteni povezanosti s podjetji in organizacijami širom po svetu. Veliki uporabniki IBMove tehnologije so tudi multinacionalke, ki imajo svoje mreže razpletene od tovarn do razdeljevalnih potrošnih mrež po svetu. V tem okviru se obvladujejo potrebe po mednarodnih podatkovnih, zvočnih in slikovnih komunikacijah. Ko IBM vzdržuje in izgrajuje vse te računalniške sisteme ter jih povezuje v mreže z dodatnimi terminali, vpliva tudi na sklepanje kupčij, s katerimi zagotavlja te razpršene operacije. Dodatni faktor, ki vpliva na sklepanje pogodb v korist IBMa, je kompleksnost vzpostavljanja in uporabe mrež prek nacionalnih meja. Vsaka dežela ima svoje predpise in iskanje napak v mednarodnih povezavah je lahko izredno težavno. Tem težavam se podjetja izognejo tako, da uporabljajo že obratujoče mednarodne mreže. IBM je zgradil svojo lastno svetovno mrežo, osnovano na SNA, ki podpira lastne daljinske operacije. Ta mreža povezuje prek 300000 terminalov, ki jih uporablja vodstvo podjetja IBM. Ta mreža je tudi hrbtenica IBMove svetovne informacijske mreže. Ta mreža se razteza prek 70 držav in povezuje IBMovske in druge naprave, ko nudi domače in mednarodne storitve.

IBMova mrežna infrastruktura podpira vrsto zahtevnih storitev in je močno podprta tudi z eksperti v IBMovskih vladnih lobijih širom po svetu. Informacijska mreža obratuje na več ravneh. Uporabniki IBMovih strojev AS400 imajo neposreden dostop do IBMovskih računalnikov, pri katerih lahko zahtevajo podporo in vzdrževanje. Na naslednji ravni omogoča ta mreža osnovne funkcije, kot so konverzija protokolov, tako da lahko uporabniki komunicirajo npr. z DECovimi miniji oziroma se ti lahko povežejo z IBMovi mainframi v okviru mreže SNA. Na naslednji stopnji so omogočene storitve, ki zadevajo upravljanje podatkov, kar omogoča podjetjem enostavno iskanje in vzdrževanje pri komunicijskih storitvah. Še višje so aplikacijske storitve, kot je elektronska pošta, elektronsko trženje, dostop v podatkovne baze in finančna informacija. Te storitve so možne od PCjev do mainframov po kabelskih mrežah širom po svetu, ki so praktično javne in lahko tudi zasebne.

Obdelava podatkov kot je konverzija protokolov, usmerjanje sporočil med računalniki in obdelava različnih transakcij se le malo razlikuje od značilnih računalniških aplikacij, ki so pod-

stat IBMove programske opreme CICS in DB2. IBMovo prodajo podjetja Rolm ni mogoče razumeti kot konec upanja za neko zbliževanje, temveč kot zorenje zamisli, kaj naj zbliževanje računalništva in telekomunikacij pomeni. IBM seveda ni potrebno, da postane vodilna sila v takem telekomunikacijskem prenosu in preklapljanju, ki predstavljata drago obliko. IBM lahko uporabi svojo tradicionalno spretnost tako, da se osredotoči na ravnino, kjer dominira manipuliranje s podatki veliko bolj kot električni tok in njegova tehnologija. To pa je v bistvu strategija, ki je edinstvena in izvirno IBMovska in ki na svojem koncu pomeni zbliževanje dveh tehnoloških in uporabniških področij.

V zadnjih nekaj letih je IBM podpisal več soglasjij o sodelovanju z nemškim Siemensom in švedskim Ericsonom na področju uporabe ISDN (Integral Services Digital Networks) in v smislu razvoja telekomunikacijske ekspertize s kooperacijo. K temu pa je treba dodati še sodelovanje z japonskim NTT in nemško Deutsche Bundespost, ki uporablja računalnike za svoj Bildschirmtext. V Franciji je IBM uokviril posel s softwarsko hišo Sema-Metra in s tremi bankami (Paribas, Credit Agricole in Credit du Nord), ki jim nudi vans za finančni sektor. Vans je najmanjši toda najhitreje rastoči sektor telekomunikacijske storitvene industrije, kjer je računalniška ekspertiza pomembnejša od telekomunikacijske spretnosti.

Napaka, ki jo je IBM naredil v svojem razmišljanju, je prepričanje, da se bo programska oprema za telekomunikacijske storitve nahajala v preklopnih vezjih, ki so se do pred kratkim imenovala telefonske centrale, ki povezujejo uporabnike. Toda ISDN ločuje signalne podatke od dejanskega klica. V novi generaciji digitalnih preklopnikov, kot je npr. britanski System X, je potrebna velika količina programske opreme za krmiljenje elektronskih preklopnikov. Toda že v naslednji generaciji programske opreme, ki je namenjena dodatnim storitvam, kot so virtualna zasebna vezja, centreksi in druge zapletene naprave, je lahko programska oprema locirana kjerkoli, npr. tudi v IBMovih računalnikih. Ti procesi morajo podpirati takšne storitve, ki se pojavljajo pod skupnim imenom inteligentna vezja. To so npr. vezja za odločanje, ki so odvisna od dnevnega časa ali pozivnega izvora in ki npr. povežejo poziv s prodajo podjetja. V to vrsto funkcij spadajo tudi različne podatkovne obdelave, procesiranje transakcij in upravljanje podatkovnih baz; to vse pa so področja, ki jih IBM ne le obvlada, temveč ima dodatno znanje, izkušnje in uporabniško bazo.

Zaradi vsega tega ostaja napoved, da bo IBM postal tudi največje telefonsko podjetje na svetu, povsem upravičena.

A. P. Železnikar

Trije trgi oblikujejo računalniško industrijo

Prvih sto računalniških prizvajalcev iz Azije, Evrope in Severne Amerike je v preteklem letu pridelalo vrednost 243 milijard dolarjev ($243bn) produktov informacijske tehnologije in storitev. Ta industrijska dejavnost se je povečala predvsem izven ZDA. Pravijo, da je računalniška industrija dosegla potrebno stopnjo zrelosti in konsolidacije. Nekoč osamljeni jezdec IBM je podpisal sporazume o sodelovanju s konkurenco. Nekatera teh podjetij so tako majhna, da so imela probleme, kako dobiti telefonsko številko Johna Akersa (predsednika IBM), toda manj težav pri njegovem podpisu na pogodbi. Proračuni za raziskave in razvoj so preživeli nevarne kirurške posege. Pojavil se je tudi že strah pred skupnim evropskim trgom, ki bo verjetno zaščiten in nove strategije, kako obdržati dohodek po letu 1992. Šibkost trga ZDA je bila in bo čedalje bolj kompenzirana z evropskim in japonskim trgom.

Pomankanje pomnilnih integriranih vezij je povzročilo določeno sušo in visoko temperaturo v informacijski industriji. In nekateri so se morali umakniti iz poslovne igre, saj uporabniki niso prenesli cen, ki so jih podjetja plačevala za pomnilna vezja.

Podatki, ki jih prinaša Datamation (15. junij 1989), izkazujejo dohodek 100 podjetij računalniške industrije (RI) za leto 1988 v višini $243,1bn in s porastom 16,3%, kar pa je manj od skoka 18,7% v letu '1987. Profiti so sicer narasli za 15,8%, vendar ta podatek zbledi pri narastku 27,2% v prejšnjem letu. To kar je popolnoma jasno, je naraščanje mednarodnih poslov. RI je postala tudi bolj globalna kot kdaj koli poprej: na to so vplivale sile trga, pretok kapitala in tehnološki imperativ.

Padec prodaje v ZDA in cenejši dolar sta pospešila mednarodno poslovno aktivnost ameriških informacijskih podjetij. Tako je podjetje IBM že drugo leto prodalo več v tujini kot doma. Še več, dejanska rast prodaje IBMa je bila dosežena zunaj ZDA. Skupaj je že 14 ameriških podjetij od stotih doseglo več kot polovico dohodka na zunanjem trgu, med njimi tudi DEC, HP, NCR, Tandem Computers in Microsoft. Podjetje Compaq pa je povečalo prodajo v Evropi za 145% in se je v Evropi uvrstilo med 25 največjih informacijskih proizvajalcev. Vendar je zanašanje na šibek dolar postalo dvomljiva opora v dolgoročnem planiranju, saj se je že spomladi l. 1989 dolar okrepil in tako so se zmanjšale tudi močnosti za prekomorsko prodajo ameriških proizvajalcev.

Evropska razvojna scena

V Evropi so znanilci unifikacije evropskega trga po letu 1992 spodbudili preobrat v meddržavnih zvezah. Evropska podjetja, ki so bila doslej prilagojena zaščiti domačih trgov, morajo hitro privzemati merila in upravljavsko iznajdljivost, da bi bila ustrezno pripravljena za novo obliko odprtosti evropskega trga.

Finsko računalniško podjetje Nokia je danes eno najaktivnejših evropskih podjetij ($1165m dohodka v l. 1988), ki je pridobilo posle sosednjega, švedskega podjetja LMEricson Telephone Co. in se razgleduje v Zapadni Nemčiji z namero, da prevzame mesto televizijskega proizvajalca. Siemens AG je sklenil strateški telekomunikacijski dogovor z italijanskim Italtel, od IBMa pa je v ZDA odkupil podjetje Rolm. Siemens si je dovolil tudi množično reorganizacijo podjetja v l. 1988, podobno kot Olivetti in s podobnim ciljem: s sploščitvijo korporativne piramide in z večjo prožnostjo tržnega reagiranja.

Britanski STC PLC, v katerem ima Northern Telecom 24%-ni delež, je prevzel od NT termi-

nalski posel in onstran Atlantika še miniračunakniškega proizvajalca Computer Consoles in operacije podjetja Datachecker v sestavi National Semiconductor Corp. Druga največja evropska softwarska in storitvena hiša je nastala z združitvijo britanskih podjetij Systems Designer in Scicon International za francoskim podjetjem Cap Gemini Sogeti. Podobno sta se okrepili in razširili svojo mednarodno dejavnost tudi italijansko softwarsko in storitveno podjetje Finsiel SpA in francosko izposojevalno podjetje Europe Computer Systems, ki je last francoske Bank Societe Generale.

Nevarnost prevelikega vztrajanja računalniškega proizvajalca na domačem trgu je dobila svoj epilog v podjetju Norsk Data AS, pri katerem se je pokazala izguba že v l. 1988, ko je to podjetje v domačem okolju izgubilo ključne strateške posle v konkurenci s podjetji Apple, DEC, IBM in Bull. Po mnenju izvedencev sta tako Norsk Data kot nemški Comparex Informationsysteme GmbH podjetji, ki se morata čimprej izviti iz prekomerne odvisnosti njunih domačih baz. Podobna ugotovitev velja tudi za jugoslovanskega računalniškega proizvajalca Iskro Delto, ki se je v preteklem in letošnjem letu sicer preusmerjala v mednarodne posle, vendar se je kljub temu znašla v podobnih težavah.

Japonci motrijo globalna tržišča

Tudi japonska podjetja se vmeščajo v novonastajajoča globalna tržišča. Do sredine osemdesetih let je bila japonska proizvodnja usmerjana na izvozne trge, ta trend pa je bil obnovljen zopet v letu 1988. Nujnost take usmeritve je bila pogojena z močjo jena in z bojaznijo trgovinskih omejitev tako na evropskem kot ameriškem trgu. Japonsko podjetje Alps Electric je napovedalo gradnjo tovarne v Zapadni Nemčiji in razširitev svojih operacij v ZDA. Podjetji Seiko Epson in Fujitsu sta napovedali nove tovarne v Britaniji, podjetje Mitsubishi Electric pa svoje zastopstvo in tovarno v Franciji.

Med prvimi 100 računalniškimi podjetji na svetu najdemo kar 17 japonskih in daljnjevzhodnih podjetij, 22 jih je iz Evrope in 61 iz ZDA. Med prvimi desetimi na svetu so tri japonska (Fujitsu, NEC, Hitachi),le dve evropski podjetji (Siemens in Olivetti) in pet ameriških (IBM, Digital, Unisys, HP, NCR). Teh deset podjetij - ob čakanju podjetja Groupe Bull, da se mednje povzpne - oblikuje novo informacijko oziroma računalniško hunto. Teh deset podjetij obvladuje več kot polovico celotnega informacijskega trga (v l. 1988), pri čemer odpade samo na IBM 22,6%. Rast teh globalnih informacijskih velikanov pa že vzbuja skrb možnosti destruktivnega trgovinskega spopada. Ta spopad se je v preteklosti pokazal kot možnost med ZDA in Japonsko, po letu 1992 pa se na spopadnem prizorišču pojavlja še Evropa.

Naraščanje globalne kooperacije

Izkušnje iz trgovinskih spopadov kot globalne konkurence med podjetji pa odpirajo tudi možnost čedalje pomembnejše usmeritve: naraščanje stopnje globalne kooperacije med danes največjimi multinacionalnimi podjetji. Skladno s tem scenarijem pa ni največji strah teh podjetij v medsebojni konkurenci temveč v nastajanju malih podjetij na njihovih domačih trgih, saj prav ta podjetja razvijajo inovativne produkte in osvajajo nenavadne deleže na trgu. Prav v okviru teh pojavov je mogoče iskati nove načine globalne konkurence med evropskimi, japonskimi in ameriškimi multinacionalkami (glej npr. K. Ohmae, Triad Power, Free Press, 1985). Danes je povsem jasno, da želijo vodstva globalnih podjetij zaščititi svoj položaj na domačem trgu z vsemi sredstvi in tudi tako, da pristajajo na konkurenco iz območja globalne triade kot prednostjo, ki jo ima sporazumevanje pred destruktivnim trgovinskim spopadom.

V zadnjem letu je rast informacijskega trga v ZDA znašala le 8,4%, medtem ko je dosegla v Evropi 14,5% in v azijsko-pacifiškem prostoru, vključno z Japonsko, 29%. Ameriška rast je upadla predvsem zaradi čedalje večje nepopularnosti velikih sistemov. Tako so prav IBM, CDC, NCR in Unisys poudarjali prav prodajo mainframov, ki je na ameriške trgu upadla. Japonci so še vedno dosegli 20% povečanje prodaje mainframov na domačem trgu, v ZDA pa je le podjetje Amdahl povečalo prodajo svojih procesorjev za 32,3%. Pričakovati je, da bo prodaja velikih sistemov še naprej upadala.

Bolehni miniji

Prodaja miniračunalnikov, vključno z delovnimi postajami, je bila proti pričakovanju nezdrava. V ZDA se je Digital dokončno slabo odrezal (oplel), dočim sta bili podjetji Wang Labs in Data General pozitivno slabokrvni. Veliki evropski (finski) miniračunalniški proizvajalec Nokia (ki je prevzel informacijske posle podjetja Ericson) in norveški Norsk Data sta imela slabo letino. V svetovnem merilu je prodaja miniračunalnikov narasla le za 8% pri rasti celotnega informacijskega trga za 10%. Vse pozornosti vredno pa je ostalo področje procesiranja sprotnih transakcij (PST), ki predstavlja že enega od dveh prodanih sistemov.

Stagnirajoče zmogljivosti mainframovskih in miniračunalniških segmentov naj bi dokazovale, da je področje informacijskih sistemov stopilo v svojo zrelo fazo. Vendar pomeni v nekaterih industrijskih segmentih, kot je npr. software, dozorevanje tudi rast, ki se bo nadaljevala. Prodaja softwara je narasla za 22,3% in je dosegla $20,8bn. Rast podjetja Oracle s 114,4% je na področju softwara edinstvena. Podjetje Oracle zelo uspešno upošteva standarde, odprto arhitekturo in pomanjševanje, s katerimi so obsedena velika podjetja. In Oracle odkriva tržne niše, ko uporabnikom pojasnjuje svojo filozofijo.

Dohodek računalniških storitvenih podjetij se je povečal za zdravih 23,1% (upoštevajoč 100 najboljših), ko so ta podjetja vstopila v področje t.i. sistemske integracije. To velja tako za Evropo kot za ZDA. Tu se skrivajo tudi največje trenutne možnosti rasti domače računalniške industrije.

Moč mikroračunalnikov

Največja pozornost se posveča še vedno področju mikroračunalnikov. Kljub občutnemu pomanjkanju pomnilniških integriranih vezij je podraja narasla za pomembnih 26,8% in dosegla če 11,7% celotne računalniške prodaje, tako da se že približuje prodaji mainframov. Podjetja Compaq, Apple, Toshiba in Zenith Electronics so v preteklem letu znantno napredovala. Podjetje Apple je v ZDA že prehitelo IBM v tržnem deležu prodaje PCjev glede na mlačen sprejem IBMovega PS/2 in njegovo prehitro umaknitev ATja. Napre-

doval je tudi Atari, ki proda 85% proizvodnje v Evropi in podjetje AST Research (s katerim sodeluje tudi naša domača industrija). Močno pa se pojavljajo tudi tajvanska in korejska podjetja. Tako je tajvanski Acer prodal že za $300m PCjev.

Trg računalniške periferije je dosegel v preteklem letu vrednost $61,8bn s prehodom na 3,5 colske diskovne naprave. Prodaja naprav za podatkovno komunikacijo, ki vključuje kominikacijske procesorje, javne centrale, modeme, multiplekserje itd. je narasla le za borih 6,7% na vrednost $16bn v letu 1988. Podjetje AT&T je investiralo $6,7bn v digitalizacijo svoje mreže; ta proces naj bi se končal do l. 1992. Podjetje Northern Telecom je investiralo $200m v rekonstrukcijo izdatkov in napovedalo svoj prodor v Evropo in Azijo zaradi sploščitve trga v Severni Ameriki.

Računalniško vzdrževanje ostaja pomemben vir dohodkov informacijskih podjetij, saj je v l. 1988 doseglo dohodek $30bn. V to kategorijo sodi tudi izposojanje računalnikov, ki pa ni več tako donosno, kot je bilo poprej. V tej povezavi so zanimivi tudi podatki o izgubarjih. Med 100 največjimi računalniškimi podjetji je tudi 10 izgubarjev, in sicer: dva proizvajalca miniračunalnikov (Data General in Norsk Data) zaradi prevelike navezanosti na domače tržišče; dva proizvajalca diskov (Micropolis in Seagate Technology), ki sta napačno ocenila hitri pomik v to tržišče itd. Računalniško podprto oblikovenje in proizvodnja (CAD/CAM) sta prav tako napredujoči področji in z njima delovne postaje, kjer je podjetje Hewlett-Packard naredilo bistven prodor, z njim pa v Evropi tudi finski Nokia (z nakupom Ericsonovih operacij).

Neuresničene integracije

Med zanimivimi povezavami, ki naj bi se uresničile, vendar se niso, so npr. tele: za NCR je bilo rečeno, da je v interesnem polju najprej podjetja Unisys potem pa AT&T. Za Apple in Cray Research se je predpostavljalo, da sta idealna partnerja, podobno se je ugotavljalo tudi za Wang in Xerox. Xerox naj bi s tem prestrukturiral svoj marketing posebno na področju izdajateljskih sistemov.

Medtem ko se združevalna obsedenost v računalništvu ni uresničila pa se je oblikoval kritičen pogled na razrahljano zaupanje dejavnosti raziskav in razvoja (R&D). Velja poudariti, da se je za raziskave in razvoj ponekod še vedno izločalo do 15% dohodka, vendar je povprečje v primerjavi s prejšnjim letom zdrknilo iz 10,9% na 9,9%. To upadanje R&D v ZDA se dogaja v žasu, ko Japonci ponovno oživljajo dejavnosti R&D na globalni ravni! NEC je odprl svoj Research Institute v Princetonu, NJ, podjetja Epson, Hitachi in Canon pa R&D enote v Evropi in ZDA. Od ameriških podjetij le IBM ni spreminjal svoje R&D strategije s svojo R&D prisotnostjo širom po svetu. Študija ustanove National Science Foundation kaže, da namenjajo ameriška podjetja več sredstev za R&D dejavnosti v tujini.

Po vsem tem je mogoče ugotoviti, da je bila ameriška strategija na začetku osemdesetih let globalna; ozadje tega je bil najhitreje rastoči ameriški trg. Danes temu ni več tako. Pobudo prevzemajo Japonci in Evropa, medtem ko je ameriško tržišče uravnoteženo.

A. P. Železnikar

Poslovni podatki za računalniška podjetja na svetu v koledarskem letu 1988

V spodnji preglednici so zbrani prihodki nekaterih računalniških podjetij po svetu za leto 1988. Ta preglednica daje razvrstitev največjih računalniških podjetij in v njo so uvrščena tista podjetja, ki bi za domačo strokovno (beri poslovno) javnost lahko bila zanimiva.

Nekaj vodilnih računalniških podjetij glede na prihodek v preteklem letu

Mesto	Podjetje	Prihodek
1	IBM	US$ 55 002,8m
2	Digital Equipment	12 284,7m
3	Fujitsu	10 999,1m
4	NEC	10 475,7m
5	Unisys	9 100,0m
6	Hitachi	8 247,6m
7	Hewlett-Packard	6 300,0m
8	Siemens (Muenchen)	5 951,0m
9	Olivetti (Ivrea)	5 427,9m
10	NCR	5 324,0m
11	Groupe Bull (Paris)	5 296,7m
12	Apple	4 434,1m
13	Toshiba	4 226,6m
14	Matsushita	3 441,0m
15	Canon	3 391,6m
16	Control Data	3 254,5m
17	Wang	3 074,4m
18	Nixdorf (Paderborn)	3 044,9m
19	NV Philips (Eindhoven)	2 794,6m
20	Xerox	2 650,0m
21	AT&T	2 445,0m
22	STC (London)	2 425,1m
23	Memorex Telex (Amsterdam)	2 078,5m
24	Compaq	2 065,6m
28	Amdahl	1 801,8m
31	Alcatel (Paris)	1 716,0m
37	Sun Microsystems	1 461,6m
41	Atlantic Computers (London)	1 341,6m
46	Inspectorate International (Neuchatel)	1 230,3m
47	Societe Generale (Paris)	1 222,6m
50	Nokia (Helsinki)	1 165,1m
53	Cap Gemini Sogeti (Paris)	976,5m
57	Econocom (Amsterdam)	897,0m
59	Amstrad (Essex)	841,8m
62	Alps	785,2m
63	Mannesmann Kienzle (Duesseldorf)	779,0m
66	Microsoft	718,9m
73	Comparex IS (Mannheim)	614,5m
77	Racal Electronics (Berkshire)	554,1m
78	Finsiel (Roma)	545,4m
82	Lotus	468,5m
83	AST Research	459,0m
85	Norsk Data (Oslo)	450,2m
90	Oracle	424,6m
92	Acer Group (Tajpeh)	379,4m
94	Sema Group (London)	375,1m
95	SD-Scicon (Hampshire)	366,4m
Skupno v l. 1988 (100 vodilnih podjetij)		US$ 243 100,0m

Zanimivo je, koliko od navedenega skupnega prihodka so realizirala posamezna geografska

področja oziroma države. Oglejmo si tole tabelo:

Podjetje/država	Prihodek od celote $243,1bn	
IBM	55 002,8m	22,626%
Japonska	53 528,0m	22,019%
Evropa	40 094,3	16,493%
ZR Nemčija	10 389,4m	4,274%
Francija	9 211,8m	3,789%
Italija	5 973,3m	2,457%
Združeno kraljestvo	5 904,1m	2,429%
Nizozemska	5 770,1m	2,374%
Skandinavija	1 615,3m	0,664%
Švica	1 230,3m	0,411%

Pri tem je treba seveda upoštevati, da v tej preglednici ni podatkov s področja Vzhodne Evrope, kjer se nahajajo nekatera velika računalniška podjetja, ki bi lahko v naslednjem obdobju bistveno preobrnila prihodkovno težišče v korist Evrope, in sicer tako finančno kot tržno / razvojno. Danes sta v Evropi podjetji Siemens in Olivetti domala prihodkovno izravnani in na računalniškem področju bi se lahko ponovila zgodba iz avtomobilske industrije (Volkswagen in Fiat). Seveda pa ne gre podcenjevati tudi drugih evropskih proizvajalcev, zlasti podjetja, kot so francoski Groupe Bull, nemški Nixdorf, nizozemski NV Philips, britanski STC, nizozemski Memorex Telex itd.

A. P. Železnikar

Izgubarji in nazadovalci v računalniški industriji v letu 1988

V letu 1988 so nekatera znana računalniška podjetja pridelala tudi izgubo:

Mesto	Mesto med 100	Podjetje	Izguba v 1988
1	21	AT&T	US$ 1 669,0m
2	98	Atari	84,8m
3	42	Data General	48,9m
4	85	Norsk Data	41,6m
5	99	Micropolis	19,4m
6	52	National Semic.	18,1m

Seveda so zanimivi tudi podatki o podjetjih, ki so nazadovala:

M	M100	Podjetje	1988	1987	%
1	31	Alcatel	ECU1 450,0m	1 780,0m	18,5
2	109	Tandon	$ 309,3m	374,0m	17,3
3	68	General Elec.	675,0m	750,0m	10,0
4	118	Gould	275,0m	299,3m	8,1
5	77	Racal Elec. P	311,4m	335,8	7,3
8	46	Inspectorate Internetional	FF1 800,0m	1 835,3	1,9

Največji prihodek na zaposlenega v računalniški industriji leta 1988

V letu 1988 so nekater računalniška podjetja, zbrana v razpredelnici, dosegla tele prihodke na zaposlenega:

Mesto med 100		Podjetje	Prihodek na zapos.	Prihodek celoten	Število zapos.
1	41	Atlantic Computers	$937,5k	$1341,6m	1431
2	57	Econocom International	$560,6k	$ 897,0m	1600
3	73	Comparex	$545,3k	$ 614,5m	1127
4	12	Apple	$409,2k	$4434,1m	10836
5	24	Compaq	$344,3k	$2065,6m	6000
6	54	Commodore	$274,0k	$ 926,1m	3380
7	25	Nihon Unisys	$270,2k	$2057,7m	7616
8	83	AST Research	$226,7k	$ 459,0m	2025
9	28	Amdahl	$217,1k	$1801,8m	8300
10	66	Microsoft	$205,3k	$ 718,6m	3500
11	82	Lotus	$182,8k	$ 468,5m	2563
12	37	Sun Microsystems	$177,1k	$1461,6m	8253
13	38	Tandem Computers	$162,9k	$1424,7m	8745
14	81	Wyse Technology	$152,7k	$ 488,7m	3200
15	71	Apollo Computer	$147,0k	$ 653,5m	4446

Zanimivo je, da dosegajo največji prihodek per capita prav nekatera evropska podjetja, tj. prvi Atlantic Computers (ZK), drugi Econocom International (Nizozemska) in tretji Comparex (ZR Nemčija). Povprečni prihodek per capita najboljših petnajstih znaša $320,8k pri 73022 zaposlenih. Tabelarično imamo:

Podjetje /skupina	Prihodek per capita	Število zaposlenih
Atlantic Computers	$937,5k	1431
prvi trije	$681.1k	4158
prvih pet	$559,2k	20994
prvih deset	$399,0k	45815
prvih petnajst	$320,8k	73022

Pri tem je morda zanimivo, kakšen prihodek per capita iz računalniških dejavnosti so dosegli največji:

M	Podjetje	Prihodek per capita	Število zaposlenih
1	IBM	$154,2k	387112
2	Digital Equipment	$ 98,8k	124400
3	Fujitsu	$116,0k	94825
4	NEC	$ 99,3k	105486
5	Unisys	$106,5k	93000
6	Hitachi	$ 56,2k	161000
7	Hewlett-Packard	$113,0k	87000
8	Siemens	$ 16,9k	353000
9	Olivetti	$ 94,3	57560

10	NCR	$ 99,8k	60000
11	Groupe Bull	$116,3	45557
12	Apple	$409,2	10836

Seveda velja omeniti, da je to le prihodek per capita iz računalništva in da niso upoštevani dohodki iz drugih dejavnosti podjetij v razpredelnici (npr. pri IBMu, Fujitsu, Hitachiju, Siemensu itd.)

A. P. Železnikar

Računalniška podjetja z največjo donosnostjo v letu 1988

Donosnost je mogoče ocenjevati z več vidikov. Oglejmo si dve vrsti donosnosti (return), in sicer donosnost iz prometa (prodaje) (return on sales) in donosnost celotnega kapitala oziroma poslovnih sredstev (return on assets).

Donosnost iz prometa:

M	M100	Podjetje	1988	1987
1	66	Microsoft	21,1%	20,4%
2	64	Cray Research	20,7%	21,4%
3	55	Computer Associates	15,4%	9,3%
4	59	Amstrad	14,9%	17,9%
5	90	Oracle	14,6%	15,0%
6	82	Lotus	12,6%	18,2%
7	28	Amdahl	12,4%	9,7%
8	24	Compaq	12,4%	11,1%
9	33	Automatic Data Processing	11,0%	10,2%
10	60	Intergraph	11,0%	10,9%
11	2	Digital Equipment	9,8%	12,4%
12	1	IBM	9,7%	9,5%
13	12	Apple	9,5%	9,2%
14	7	Hewlett-Packard	8,3%	8,0%
15	97	Diebold	7,9%	8,1%

Donosnost celotnega kapitala oziroma donosnost poslovnih sredstev:

M	M100	Podjetje	1988	1987
1	41	Atlantic Computers	31,3%	52,7%
2	59	Amstrad	30,4%	46,6%
3	66	Microsoft	25,0%	32,3%
4	91	Science Applications	24,0%	8,4%
5	90	Oracle	18,5%	16,0%
6	12	Apple	18,4%	17,3%
7	24	Compaq	16,1%	15,1%
8	64	Cray Research	15,8%	16,3%
9	82	Lotus	14,0%	22,7%
10	55	Computer Associates	12,3%	7,7%
11	2	Digital Equipment	11,8%	13,6%
12	78	Finsiel	11,7%	15,6%
13	28	Amdahl	11,6%	9,7%

Bilo bi koristno, če bi se pri zadnjih dveh razpredelnicah zamislili tudi naši ekonomisti in finančniki, ko bi na podoben način ocenjevali donosnost domačih podjetij, še zlasti po številnih reorganizacijah v naši elektronski industriji.

A. P. Železnikar

Računalniška podjetja z največjo rastjo prihodka v letu 1988

Čeprav se pri nas lahko čudimo vsakršnji ekonomski rasti domačih podjetij pa nekatera računalniška podjetja v razvitem svetu izkazujejo prav to, čemur bi pri nas rekli prihodkovna megalomanija. Seveda ni mogoče kar tako odgovoriti, kako jim to uspeva. Petnajst prihodkovno najbolj rastočih računalniških podjetij je zbranih v naslednji preglednici:

M	M100	Podjetje	prih. 1988	prih. 1987	%
1	50	Nokia	Fmk4877m	1835m	165,8
2	88	Nynex	$ 430m	180m	138,9
3	90	Oracle	$ 425m	198m	114,4
4	37	Sun Microsystems	$1462m	756m	93,4
5	94	Sema	FS 211m	121m	74,8
6	67	Continental Info.	$ 698m	405m	72,3
7	24	Compaq	$2066m	1224m	68,7
8	74	Miniscribe	$ 603m	363m	66,4
9	34	Prime Computer	$1594m	961m	65,9
10	49	Arthur Andersen	$1199m	749m	60,0
11	83	AST Reasearch	$ 459m	290m	58,4
12	66	Microsoft	$ 719m	457m	57,4
13	12	Apple	$4434m	3041m	45,8
14	98	Atari	$ 362m	250m	44,7
15	55	Computer Assoc.	$ 925m	649m	42,6

Finski Nokia in britanski Sema sta evropska rekorderja v prihodkovni rasti, razvidno pa je tudi, kako se dobro drčijo znani mikroračunalniški proizvajalci (Compaq, Apple in Microsoft). Kdaj bomo lahko začeli zbirati podatke o prihodkovni rasti naših računalniških podjetij tudi pri nas (seveda na dolarski ali markini osnovi)?

A. P. Železnikar

IBM v preteklem letu

Za predsednika podjetja IBM Johna Akersa (IBM CEO) pravijo, da je postal partner partnerjev.

IBM (Old Orchard Road, Armonk, NY 10504) je v preteklem letu dosegel računalniški prihodek 55 002 800 000 dolarjev, in sicer 42% v Severni Ameriki, 36% v Evropi, 15% v Aziji in na Pacifiku in 7% drugje. Značilnost njegove politike je bila partnerstvo s tekmeci. Če je bilo leto 1987 leto uporabnikov, je bilo leto 1988 leto partnerjev. Proti strategiji AT&T je IBM reagiral skupaj z Digitalom z medpodjetniškim konzorcijem za Unix. Svoja zaščitena integrirana vezja je ponudil tako miniproizvajalcu Digitalu kot prizvajalcu mainframov Siemensu, ki je v Evropi glavni tekmec IBMa. S Siemensom je sklenil skupne marketinške posle, pri čemer je Siemensu prodal telekomunikacijsko podjetje Rolm. Drugi bistven dogovor je bil sklenjen z bivšim predsednikom podjetja Apple glede pravic uporabe uporabniško prijaznega vmesnika podjetja NeXT Computer.

Prijateljske overture so bili deležni tudi

Japonci, saj je bila sklenjena vrsta poslov z Nippon Steel, Nissan in Sumitomo Electric Industrial. Japonci lahko odslej kupujejo tudi IBMove mainframe s kloniranimi japonskimi operacijskimi sistemi.

IBMova mreža zvez in koalicij je zdaj že tako kompleksna, da je ni mogoče več enostavno pregledati. Svetovna mreža preprodajalcev z dodano vrednostjo (VARS je kratica za value-added resellers) in tržnih partnerjev obsega že tisoče partnerjev in še narašča. Seveda pa je IBMov najljubši partner še vedno uporabnik. IBMova značilnost je sposobnost podpore širokega spektra aplikacij vse do industrijskega know-howa. V prejšnjih letih je bila namreč ta ekspertiza zanemarjena zaradi okostenele birokracije in nekoristnih produktov. V letu 1988 je prišlo do obrata, in sicer v odnosih IBMa z njegovimi uporabniki in v njegovih produktih.

IBMove operacije v ZDA so bile razceplene v pet poslovnih linij (prav za prav v šest, če se upošteva še obnovljena marketinška organizacija), vsaka z decentraliziranim odločanjem in produktno odgovornostjo. Tako si je IBM ponovno pridobil zgubljeni področji mainframov in miniračunalnikov. Njegov 103 MIPSni plus model z oznako 3090S je vrnil IBMu prvo mesto podjetja v železarski ligi. Toda model AS/400 je postal eden najuspešnejših produktov v novejši zgodovini IBMa, kot je izjavil John Akers; ta model je dobesedno zasvojil uporabnike. IBMova strategija vsakršnje rešitve je tako v povezani ponudbi stroja, podatkovne baze, aplikacije in uporabniške prijaznosti. Tako je IBM izdobavil kar 30000 teh sistemov v drugi polovici leta 1988 in realiziral višek več kot treh milijard dolarjev v šestih mesecih takorekoč iz nič.

IBM naj bi ponovno osvojil tudi mišljenje uporabnikov na področju osebnih računalnikov. Na pohodu je večja pripravljenost, da se uveljavita IBMov operacijski sistem OS/2 in mikrokanalsko vodilo. Na potezi so tudi proizvajalci iz klonskega konzorcija. IBMovi posli v ZDA v primerjavi z lanskim letom niso bistveno napredovali zaradi forsiranja prodaje železnine namesto človeških zmogljivosti. Toda to se je v začetku letošnjega leta že spremenilo.

Partnerstvo, popravljanje preteklih napak in izločanje obrobnih operacij so značilnosti IBMa v preteklem letu. Dobiček podjetja je narastel za zdravih 10,4% na 5,81 milijard dolarjev, prodaja pa se je povečala za 8% in se približala skupnim (okroglim) 60 milijardam dolarjev.

A. P. Železnikar

Digital Equipment v letu 1988

Podjetje Digital Equipment Corp. (146 Main Street, Maynard, MA 01754) je v preteklem letu realiziralo prihodek 12 284 000 000 dolarjev, in sicer 48% v Severni Ameriki, 36% v Evropi, 15% v Aziji in na Pacifiku in 1% drugje. DEC ni več to, kar je bil v poznih sedemdesetih letih, namreč razburkano in samozavestno podjetje. Danes je DEC defenzivno podjetje, katerega vedenje razveljavlja njegovo obnašanje iz obdobja pred več kot desetimi leti. Ta opomba je prav gotovo aktualna tudi pri presojanju razmer, povezanih z domačo računalniščko industrijo, za katero je prav gotovo značilna defenzivnost na vsakem koraku.

Pred začetkom zmagovitega obdobja računalnikov VAX je obstajalo kar nekaj tekmecev DECa, ki so imeli hitrejše in zmogljivejše, tudi 32-bitne računalnike, v primerjavi z DECovo glavno usmeritvijo tipa PDP-11. Tako se je lahko razvilo podjetje Prime Computers na račun DECa, katerga prvi VAX se je pojavil nekaj let kasneje. Vendar v poslednjih letih spet oživlja spomin na vzpon DECa z njegovo VAX družino.

V letu 1988 je DECa resneje zaskrbelo hitro napredovanje skupine mlajših podjetij. Vsa ta podjetja so nastala v osemdesetih letih in se hitro polastila trga delovnih postaj in programske opreme: Compaq Computer, Novell in Sun Microsystems. DEC pa se je zavedel svoje defenzivnosti tudi v spopadu z IBMom. Čeprav še vedno lahko vstopa na trg procesiranja transakcij in storitev sistemske integracije pa mora vendarle zavarovati osvojene pozicije v inčeniringu in sistemih delovnih skupin proti IBMu. Ti novi pojavi in tržne sile so primorale DEC, da temeljito revidira svoje načrte ene same računalniške arhitekture. Tako je DEC skupaj s podjetjema Mips Computer Systems in Tandy pripravil vrsto delovnih postaj z operacijskima sistemoma Unix in MS/DOS, da bi lahko konkuriral podjetjema Compaq in Sun. V razredu cenenih mrež tipa PC LAN pa naj bi se spopadel s podjetjem Novell.

Še vedno pa DECov zamegljeni marketinški pogled ni povzročil zanikanja njegove usmeritve na področju komercialnih sistemov, tako da je DEC razkril proizvodnjo novih visoko zmogljivih računalnikov tipa VAX in programske opreme za procesiranje transakcij in s tem pritisnil na IBMovo usmeritev podatkovnih centrov. Zato je bil koncept VAXa izpopolnjen in razširjen v več smereh. Tako so nastale ločene računalniške družine za področja malih, srednjih in velikih računalniških sistemov.

Kljub takim poslovnim potezam pa je bila prodaja serije VAX 8800 pod pričakovanji in DEC je iztržil doslej najnižji dobiček oziroma njegovo rast. Profit je znašal le borih 6% ali samo $1,21bn kljub 18% porastu prodaje. Izvenameriška prodaja se je prvič povzpela na 52,4% v celotnem letnem prihodku $12,285bn.

Glede na uspešnost poznih sedemdesetih let se je v letu 1988 prav gotovo pojavil vzrok za deja vu: DEC je razkril svoje načrte nove računalniške arhitekture, ki se imenuje Application Integration Architecture (AIA). Nosilec programskega okolja te arhitekture, ki je povezana z možnostmi heterogenega računanja in industrijskih standardov, je VAX. Pri tem je pomembno, da se z njo uvršča DEC v svet namiznega in značilno unixovskega pristopa svojih mlajših konkurentov.

A. P. Železnikar

Fujitsu v preteklem letu

Japonsko podjetje Fujitsu Ltd. (Maronouchi Center Building, 6-1 Maronouchi 1-chome, Chiyoda-ku, Tokyo 100) je v preteklem letu realiziralo prihodek 10 999 100 000 dolarjev, in sicer 86% v Aziji in na Pacifiku, 10% v Severni Ameriki in 4% v Evropi. V preteklem letu je podjetje prekoračilo dva sistemska mejnika:

pridobilo si je sloves najhitrejšega računalnika (le za kratek čas) in sprejelo tržno odločitev o spremembi sklepanja sistemskih poslov. Pri tem se je predhodno oskrbelo še z ekspertizama podjetij Sun Microsystems in Amdahl o razvoju polprevodnikov in centralnih procesorjev. To sodelovanje z ameriškima partnerjema je omogočilo dobavo novih sistemov za japonski trg.

Tako je Fujitsu lahko vstopil na hitro rastoči japonski trg inženirskih delovnih postaj z licenco Sun Microsystems za SPARC (tj. riscovska Scalable Processor Architecture). Malo kasneje je Fujitsu uvedel na Japonskem Amdahlova sistema 5990-1400 in 5990-700. Ta dvojica, ki konkurira IBMovim mainframom ES/3090, je Fujitsov prvi, ibmovsko programsko polno kompatibilni par mainframov. SPARCovska družina S delovnih postaj Fujitsa pomeni tudi prvi vstop podjetja v posle Unix inženirskih delovnih postaj. Ta plasma se je na Japonskem povečal za 250% v letu 1988 in dosegel prodajo 70000 enot v primerjavi z 20000 enotami v letu 1987.

Poslovna odločitev Fujitsa kaže na nove napore v smeri revitalizacije Fujitsove usmeritve na področje mainframov. Čeprav je prihodek iz računalniške dejavnosti podjetja narastel za 11,5% in dosegel vrednost ¥ 1,4 trilijone ($11bn), je bila ta rast manjša kot v prejšnjih dveh letih. Fujitsovo računalništvo je namreč doseglo nižjo raven prihodkov kot znaša ta raven za podjetje kot celoto.

V decembru 1988 je Fujitsu začel prodajati osem modelov novo razvitih superračunalnikov serije FACOM VP 2000. Model VP 2600 zmore hitrost 4 GFLOPS (štiri milijarde operacij s plavajočo vejico v sekundi) vektorskega procesiranja. Vendar je bila najava te hitrosti kot največje na svetu le kratkotrajna, ker je NEC takoj za tem postavil na trg še hitrejši računalnik.

V vrsti mednarodnih kooperacij velja omeniti operacije Fujitsa, ki zadevajo glasovno in podatkovno komunikacijo v povezavi s podjetjem Business Communication Systems v Anaheimu v Kaliforniji. Ta operacija združuje razvoj, proizvodnjo, prodajo in storitve pod enotno organizacijo.

Fujitsu je odprl tudi svoj novi tehnološki center za raziskave in razvoj in prikazal njegovo delo na razstavi Fujitsu Technology '88. Tu je Fujitsu pokazal tudi nevroračunalniške robote in sisteme umetne inteligence in sisteme za komunikacijo s koherentnimi svetlobnimi valovi, ki lahko multipleksirajo na tisoče slikovnih in milijone zvočnih signalov. Nadalje je pokazal fotonski preklopni sistem z obsegom 512 megabitov na sekundo, napredek v tehnologiji komponent vključno s 4-bitnim mikroprocesorjem, ki temelji na Josephsonovem spoju in tudi delovanje biosenzorja, ki je sestavljen iz biološke substance.

A. P. Železnikar

Siemens v letu 1988

Zapadno nemško podjetje Siemens AG (Wittelsbacherplatz 2, D8000 Muenchen 1) je v preteklem letu realiziralo prihodek 5 951 000 000 dolarjev, in sicer 89% v Evropi, 10% v Severni Ameriki in 1% v Aziji in na Pacifiku. Siemens je največji evropski domorodni oskrbovalec, ki je svoj računalniški izkupiček v letu 1988 povečal le za suha 2% v primerjavi s 7,7% v letu poprej. To upadanje prihodka, razmehčanost evropskih poslov pisarniških sistemov in perspektiva enotnega evropskega trga po letu 1992 so v tradicionalno previdnem nemškem podjetju sprožili nove poslovne akcije. Tako je Siemens doživel svoj največji pretres na področju visokega managementa po letu 1969, da je lahko sprožil agresivno kampanjo ekspanzije v mednarodne operacije in v posodobitev svojega zastarelega svetovnega programa podpore svojih multinacionalnih računalniških partnerjev.

Izjemi na področju novih produktov sta bili le prodaja delovnih postaj (osnovanih na SINIX) in laserski tiskalniki. Vendar se je bistveno izboljšala prodaja Siemensovih osebnih računalnikov, in sicer za približno 50%. Tržno težišče je bilo v letu 1988 na sistemih Hicom PBX (to je bilo Siemensovo leto Hicoma) in se je tudi bogato izplačalo. Siemens je prodal 750 teh sistemov.

Pridobivanje novih poslov, združevanje in skupni posli so dejavnosti, ki so se uvrstile visoko v program novega Siemensa. Njegov predsednik in glavni izvršilni uradnik Karlheinz Kaske je izjavil, "da bo upošteval vsakršnjo priložnost, ki se bo ponudila, če se bo ta vključevala v strateški koncept podjetja." To pa je seveda vse kaj drugega, kot smo vajeni pri nas doma, kjer se pobuda duši, etiketira in omalovažuje.

Kot del svoje ekspanzije je Siemens združil moči z londonskim elektronskim velikanom General Electric Co. (GEC) v novembru 1988 z namero, da si pridobi tudi britanskega komunikacijskega specialista Plessey Co. (PLC). To pa pomeni, da bo Siemens lahko izboljšal svoj položaj na dobro branjenem britanskem elektronskem tržišču. V lanskem decembru pa je Siemens presenetil industrijo z najavo skupnih poslov z IBM na PBX področju, ki je bilo zgrajeno okoli IBMove podružnice Rolm, katero je IBM kupil v letu 1985. Siemens je pobral razvoj in proizvodnjo podjetja Rolm in si s tem podaljšal svojo tržno roko v ZDA, hkrati pa dosegel, da bo IBM njegov Hicom sistem prodajal skupaj z računalniško opremo v Evropi.

Drugi skupni posli Siemensa v preteklem letu zadevajo Intel, s skupno kreacijo novega podjetja BiiN v Hilboro, Oregon. To podjetje bo razvijalo misijsko kritične sisteme v ZDA in v Zapadni Nemčiji. V okviru svojega prestrukturiranja je Siemens začel menjavati delovna mesta tisočim svojih uslužbencev skladno z novo filozofijo podjetja, ki predvideva 15 do 20 prožnih in neodvisnih poslovnih skupin. Medtem pa je Siemens zmanjšal tudi svojo delovno silo za 6000 delavcev v Nemčiji.

A. P. Železnikar

Prestrukturiranje podjetja Olivetti glede na novo realnost

Podjetje Olivetti (Via G. Jervis 77, 10015 Ivrea, Italija) se je preoblikovalo s ciljem, da ohrani svojo donosnost in da se pripravi za leto 1992. S svojim visokim devetim mestom, takoj za nemškim Siemensom, je podjetje v preteklem letu realiziralo prihodek 5 427 900 000

dolarjev, s plasmajem 81% v Evropi, 10% v Severni Ameriki, 6% v Aziji in na Pacifiku in 3% drugje. Stisnjeno v ožajoče pogoje, spodbujeno s trenutno donosnostjo in ujeto v tržne situacije, ki postajajo čedalje bolj specifične, je italijansko podjetje Ing. C. Olivetti & Co. SpA v letu 1988 napovedalo, da je sprožilo masivni program prestrukturiranja. V povezavi s to reorganizacijo podjetja je bil odpoklican iz ZDA Vittorio Cassoni, ki ga je Olivetti posodil podjetju AT&T za vodenje njegove divizije podatkovnih sistemov (od novembra 1986). Tako je Cassoni postal soupravljavski direktor predsednika podjetja Carla de Benedettija.

Da bi lahko konkuriral podjetju Pacific Basin in Japoncem, je Olivetti ustanovil novo divizijo Olivetti Office za produkte široke potrošnje, kot so pisalni stroji, tekstovni procesorji in naprave za faksimile in kopiranje.

Olivettijeva sistemska in mrežna skupina se je spopadla z Digitalom, IBMom, NCRom in drugimi na področju aplikacijskih rešitev, ki zadevajo porazdeljeno procesiranje podatkov. Produktna linija skupine obsega široko področje PCjev in sistemov z delovnimi postajami, minijev, mrež in Hitachijevih mainframov, ki so IBMovsko kompatibilni, in sicer za italijansko in špansko tržišče.

Olivettijeve informacijske storitve so bile oblikovane za dejavnosti sistemske integriracije in storitev in naj bi konkurirale podjetjem, kot so Electronics Data Systems Corp. in francoski Cap Gemini Sogeti.

Motivacija za pregrupiranje Olivettija je bilo poslabševanje profitnosti med letom, ki je upadla na koncu leta na 11,5% oziroma na 356 milijard italijanskih lir ($273,4m). Prihodki podjetja v celoti so narasli za 14% na It lir 8 407 milijard ($6,5bn), pri tem so računalniški dohodki narasli za 17%. Dobiček je v preteklem letu upadel za 30%. Kljub temu so vsa produktna področja izkazovala naraščanje. Olivetti je imel težave pri gradnji svoje miniračunalniške prodaje, kljub veliki izbiri produktov, vključno s svojo lastno LSX linijo, AT&T 3B sistemi in sistemi, občutljivimi na napake podjetja Stratus Computer Inc. Prodaja miniračunalnikov se je povečala le za 2%, tj. na vrednost $614m.

Olivetti si je v letu 1988 prizadeval za integracijo svojih produktov s sistemi drugih proizvajalcev, ki so del njegove OSA (Open System Architecture). Olivetti in Digital sta se sporazumela za izmenjavo tehnologije, ki omogoča integracijo Olivettijevih PCjev v Digitalove mreže z uporabo DECovega NAS (Network Application Support). Povezava z DECom se je tako bistveno okrepila, letos pa je temu sledil še sporazum, da bo Olivetti oskrboval DECa s PCji na evropskem tržišču.

Olivettijeva skupina za sisteme in mreže je predstavila novo serijo PCjev P500 in P800, ki uporabljata Intelov procesor 386SX in IBMovo MicroChannel arhitekturo. Olivetti in ameriški Stratus sta se dogovorila za razvoj na napake občutljivega operacijskega sistema Unix za Stratusove, na napake občutljive računalnike, tako da bodo ti kompatibilni z Olivettijevimi sistemi.

Olivetti ima za seboj dolgo zgodovino investiranja v mala tehnološka podjetja širom po svetu. V svojem portfolio ima trenutno 240 podjetij, vključno z Acorn Computers v Britaniji, Bunker Bamo v ZDA, skandinavskim Scanvest-Ring in Triumph Adler v Zapadni Nemčiji. Glavna naloga Cassonija je, da poveže te ločene elemente in oblikuje koordinirane produktne linije skozi podjetje. Novo prestrukturiranje podjetja in poudarek na OSA naj bi olajšala izvedbo te naloge. Cassoni pa mora upoštevati predvsem hitre spremembe na evropskem trgu PCjev, čedalje večje oddaljevanje podjetja AT&T od Olivettija in njegovo odločitev, da si poišče dobavitelja PCjev v ZDA.

Seveda pa se približuje tudi leto 1992. To pa pomeni še kaj več kot je lahko notranje prestrukturiranje podjetja, ko mora biti to usmerjeno v unificirani evropski trg. Ta izziv pomeni prestrukturiranje celotne industrije. Cassoni dodaja k temu še, da je podobnih podjetij enostavno preveč.

A. P. Železnikar

Francoski Groupe Bull v letu 1988

Francosko računalniško podjetje Groupe Bull (121 Avenue de Malakoff, 75116 Paris) je v letu 1988 realiziralo prihodek 5 296 700 000 dolarjev, in sicer 59% v Evropi, 40% v Severni Ameriki in 1% v Aziji oziroma na Pacifiku. V francoskem podjetju Groupe Bull ima francoska vlada 92% holding. V preteklem letu je to podjetje realiziralo trdno kontrolo z globalno posestjo podjetja Honeywell Inc. Rezultat tega je, da je eno prvih ameriških računalniških podjetij v lasti francoskega ljudstva.

Bull je povečal svoj delež v ameriški podružnici Honeywell Bull Inc. iz prejšnjih 42,5% na 65,1% v letu 1988, znižal Honeywellov delež na 19,9% in pustil nespremenjen delež NECa na 15%. Ameriško podjetje je preimenoval v Bull HN Information Systems Inc. S tem je Bull lahko najavil svoj svetovno konsolidirani prihodek z vrednostjo FF 31,5 milijard ($5296,7m). Francoski del v tem prihodku ni bil večji od 39,9%. Strategija Bulla v naslednjih letih ni v visoki donosnosti, temveč v povečevanju prihodka, čeprav se je njegova donosnost v letu 1988 povečala tako v Evropi kot v Ameriki.

Pomemben nov produkt Bulla v letu 1988 je prav gotovo poslovni sistem DPS 9000, ki ga poganja operacijski sistem GCOS 8. Bull je najavil tudi mali poslovni sistem DPS 4000 in izboljšavo popularnega departmentnega sistema DPS 7000. Vstopil je tudi v področje odprte sistemske mreže z desetimi Unix sistemi, vključno z DPX 1000, 2000, 3000 in 5000, ki so bili razviti v Franciji. V začetku leta 1989 si je Bull kot prvi na svetu pridobil pravico do uporabe paketa X/Open v svoji verziji SPIX Unixa System V.3.

Na spodnjem koncu svoje procesorske ponudbe je Bull razširil svojo družino delovnih postaj Questar in uvedel svoj OS/2 mikrosistem, imenovan Micral 75. Prodaja obeh produktnih linij je v zadnjem letu izredno narasla.

A. P. Železnikar

Kovanje denarja z delovnimi postajami ali Sun Microsystems v letu 1988

Blešk podjetja Sun Microsystems se skriva v agresivnem določanju cen in visokih zmogljivostih njegovih produktov. Sun Microsystems Inc. (2550 Garcia Avenue, Mountain View, CA 94043) je reraliziralo 1 461 600 000 dolarjev v letu 1988, in sicer 60% v Severni Ameriki, 24% v Evropi in 16% v Aziji in na Pacifiku.

Ko bo zares nastopila doba odprtih sistemov, bodo proizvajalci informacijske tehnologije soočeni z igro novih pravil. Ko bodo uporabniki že zamreženi v specifičen hardware in software, bodo proizvajalci lahko pospeševali le še inovacijo in sprejemali nižjo profitno stopnjo, če bodo želeli ostati v tekmi s trgom.

Izgleda, da se vsaj eden od proizvajalcev obnaša tako, kot da je nova realnost odprtih sistemov že pred nami. Podjetje Sun Microsystems je združilo vratolomno hitrost uvajanja novih produktov z agresivnim določanjem cene, z uresničevanjem odprtih sistemov in s tem, kar je bilo poimenovano kot mrežno računalništvo. Posledica tega je bila osupljiva rast. V pičlih sedmih letih je Sun vzcvetel v milijardno podjetje ($1,4616bn) in postal zvezda na trgu delovnih postaj, vrednem $4,1bn.

Samo v letu 1988 se je prihodek Suna povečal za 93,4%, profit pa za 86,7%, ko je dosegel vrednost $89,6m. Na koncu leta 1988 je imel Sun v zakupu že 28,3% svetovnega trga delovnih postaj, tj. za 4,2% več kot v letu 1987. Ta tržni delež je Sun odvzel svojim tekmecem, in sicer podjetju Apollo Computer, katerega delež je padel za 4% na 13,5% celotnega tržnega deleža in IBMu, ki je obdržal le še 2,6% trga. Tako je porinil Apollo v naročje Hewlett-Packarda v začetku tega leta.

Sunu je uspelo znižati zapreke v postavljanju cene in zviševanju zmogljivosti ter z vnovčevanjem na zahtevo tehničnih uporabnikov in z naraščanjem števila komercialnih uporabnikov. Podobno kot osebni računalniki nudijo tudi Sunove delovne postaje podporo za vežopravilnost (multitasking) in večuporabnost, vključno z dostopom v skupne zbirke, visoko resolucijo, hitro grafiko itd. S temi lastnostmi so Sunovi produkti postali prvaki na področju programskega iženiringa in oblikovalne avtomatizacije. Ta trčni segment je pokril 68% Sunovih uporabnikov v letu 1988.

Kasneje pa so Sunove delovne postaje postale popularne tudi med manj-tehničnimi uporabniki, kot so varnostna in izdajateljska podjetja. Koncept distribuiranega, mrežnega računalništva z uporabo delovnih postaj in storilnikov (servers) je osvojil domišljijo uporabnikov Fortune 2000. T.i. namizna revolucija, ki se je začela s PCji v letu 1980, je dozorela v zahtevi po moči in odprtosti sistema v obliki zamreženih (v mrežo povezanih) delovnih postaj.

V prepričanju, da so delovne postaje naslednja etapa v namizni revoluciji, je Sun v poslednjih letih agresivno povečeval njihovo moč ob istočasnem zniževanju njihove cene do ravni osebnih računalnikov. Sun, ki je v letu 1988 uvedel delovne postaje s svojim RISC procesorjem (SPARC), je to arhitekturo dobro izkoristil. Pri tem je uvedel spodnjo verzijo s 7 milijoni ukazov na sekundo (MIPS) svoje sparcovske produktne linije Sun 4 z osnovno ceno $19950. Zgodaj v tem letu se je pojavila 12,5-mipsna verzija z osnovno ceno $8995. Letos naj bi Sun plasiral 90000 sparcovskih delovnih postaj.

Očitno bo Sun nadaljeval s svojim načinom odprtih sistemov in s strategijo agresivnih novih produktov in postavljanja cene. V letu 1988 je namenil Sun 13% prihodka raziskavam in razvoju. Tudi letos bo Sun namenjal R&D podoben delež, ki je nad industrijskim povprečjem (9,9%). Sunovo agresivno postavljanje cen je dopuščalo nizko operativno maržo (mejo rentabilnosti), ki je znašala 10%.

A. P. Železnikar

Finska Nokia na pohodu

Finska Nokia Corp. (P.O.Box 226, 00101 Helsinki), 50. računalniško podjetje v svetovni razvrstitvi, je v računalništvu realiziralo prihodek 1 165 100 000 dolarjev, predvsem z združitvijo stagnirajoče švedske podatkovno sistemske divizije z hitro rastočo finsko podružnico v enotno, dinamično in učinkovito evropsko podjetje. Prav to je uspelo njenemu briskantnemu in ambicioznemu predsedniku (Kalle Isokallio).

Nokia je rezultat združitve med Data Division of Ericson Information Systems, ki je podružnica velikega skandinavskega telekomunikacijskega podjetja LM Ericson in računalniških prodajnih operacij finskega industrijskega konglomerata Nokia. Ericson je obdržal le 20% delež, Nokia pa odločujoči vpliv.

Združitev je katapultirala prihodek Nokia Data za 166%, pri tem je profit narastel le za suhih 11,3%, vendar se je združitev Nokie pokazala kot skupinska profitnost v konglomeratu. Nova Nokia Data prodaja predvsem terminale, PCje, delovne postaje in sisteme za skupinsko delo. Tako je bilo instaliranih vež kot 700 000 profesionalnih delovnih postaj. Digital in IBM kupujeta njene terminale in prodajne monitorje za evropsko tržišče. Na Finskem distribuira Nokia tudi sisteme DPS 8 in DPS 6 francoske Groupe Bull, na napake občutljive računalnike podjetja Tandem in nekatere z IBM kompatibilne mainframe podjetja National Advanced Systems.

Uporabniška baza podjetja Nokia Data obsega bančništvo, zavarovalništvo, prodajno in javno administracijo. Zunaj Skandinavije sta njena uporabnika nemška Bundespost in britanska Midland Bank. Eden od vzrokov za nakup Ericssonovih operacij je bila prav uporabniška baza.

Produktni razvoj Nokie je pogojen s kombinacijo mreženja in ergonomije. S stališča uporabnika to ni svet OS/2 ali Unixa, temveč svet lokalne mreže, zatrjuje Isokallio. Oblikujemo sisteme, ki povečujejo uporabnost mrež s priključevanjem okolij OS/2 in Unix. Različni bomo v uporabnosti terminalov in sistemov z delovnimi postajami in to ne le na ravni radijske emisije in fizične in softwarske ergonomije, pravi Isokallio.

A. P. Železnikar

novice in zanimivosti news

Zakaj se inženirji zavzemamo za tehnološko tržno združevanje

Vedno pogosteje imamo priložnost, da preko dnevnega časopisja zvemo o krovnih ali "holding" podjetjih. Bistvo teh je povezovanje s pomočjo ustanovitve krovnega podjetja, kamor se prenese poslovni sklad. Krovno podjetje pa nato prenese ta kapital nazaj na matična podjetja. Na ta način bi družbena lastnina dobila svojega lastnika. Mnogo je tudi govora o zdravem načinu poslovnih vezi med podjetji.

Ideja je na prvi pogled imenitna, samo da se ne ve, na kakšen način se podjetja združujejo, še manj pa na kakšen način in kaj se proizvaja. Načinov združevanja je več glede na motiv združevanja. Najpogostejša je povezanost s tržnim motivom. Tako imenovana tržna proizvodna povezanost je po mojem mnenju enorazsežna in nepopolna, ker ne pove ničesar o notranjih tehnoloških pogojih za nastop na trgu.

Vsi mediji pritrjujejo, da so potrebni ekonomisti, ki znajo izračunati in upravljati s profitom, zato je motiv tržni. Da je poleg tega proizvodni, je vsakomur jasno, ker morajo profit delavci proizvesti. Nihče pa si iz prastrahu pred inženirji ne upa niti omenjati, da je vsaka proizvodnja plod izumljanja, to pa je delo inženirjev (iz fr. ingenieur = izumitelj in lat. ingenium = izum). Imamo lahko še tako popolne tržno proizvodne modele, vendar brez inženirskega, ki izumlja in tehnološkega dela, ki zna izum prenesti v proizvodnjo in naučiti delavce proizvajanja, nimamo ničesar tržiti, še manj pa profitirati.

Inženirju ni mogoče ukazati nek izum, ker je kreativnost obratno sorazmerna s stopnjo represivnosti, pač pa mu je potrebno nuditi pogoje za izumljanje, kot so laboratorijska in informacijska sredstva. Inženirju je potrebno pustiti "liberte creatrice" seveda z nekim rokom, v katerem se lahko seznanja z novostmi na svojem področju in neovirano eksperimentira. Po tem času mora imeti proizvod, ki je laboratorijsko preizkušen in pripravljen za tehnologijo in proizvodnjo. Šele potem lahko nastopa podjetje s proizvodom, ki je rezultat znanja, na konkurenčnem trgu.

Popolnejša od tržno proizvodne je tehnološko tržna povezanost. Tehnologija (iz gr. techne = spretnost, logos = razum) pove, na kakšen način proizvajati izume inšenirjev, trg pa ne določa proizvodnje, pač pa na izoblikovano proizvodnjo samo zahteva količino.

Sodeč po literaturi, je tehnološko tržna povezanost osnovana na načelu, da mora biti vsako podjetje profitno, ali pa mora slediti profitnemu podjetju. Podjetje, ki ni profitno, ni nujno tudi nekoristno, ker ga potrebujejo profitna podjetja za svoj tehnološki proces. Lahko pa je nasproti temu neko podjetje profitno, vendar nekoristno za združitev, ker kupuje proizvode za združitev nekoristnih podjetij. Dejstvo, ali je neko podjetje profitno, je določeno s tehnološkim faktorjem q(i,j). To je vrednost proizvoda podjetja j, ki ga mora podjetje i od njega kupiti, da proizvede enoto vrednosti (npr. 1$). Tehnološki faktor je vedno manjši od 1, še več, če je podjetje profitno, potem mora biti tudi vsota tehnoloških faktorjev manjša od 1.

Predpostavimo, 6 podjetij v medsebojnem tehnološko tržnem odnosu. Naj matrika Q predstavlja tehnološko matriko združitve podjetij:

$$Q = \begin{bmatrix} 1/2 & 0 & 1/4 & 0 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 \end{bmatrix}$$

Vsaka vrstica vsebuje tehnološke faktorje enega podjetja. Na prvi pogled lahko ugotovimo, da so profitna podjetja 1,2 in 6. Vsa druga so neprofitna, ker imajo vsoto vrstice enako 1.

Predpostavimo zdaj, da podjetja svoj presežek shranjujejo preko krovnega podjetja v banko. Zato se matriki Q doda še en stolpec, ki določa, koliko profita posamezna podjetja prispevajo. Vrednosti tega stolpca so enake preostanku vsote vrstice do 1. Dobimo verjetnostno matriko P, ki določa najbolj verjeten tehnološki odnos med podjetji vključno s krovnim podjetjem P(0):

$$P = \begin{array}{c} \begin{matrix} P(0) & P(1) & P(2) & P(3) & P(4) & P(5) & P(6) \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 1/2 & 0 & 1/4 & 0 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 \end{bmatrix} \end{array} \begin{matrix} P(0) \\ P(1) \\ P(2) \\ P(3) \\ P(4) \\ P(5) \\ P(6) \end{matrix}$$

Zdaj lahko ugotovimom koristnost podjetij. Krovno podjetje ne proizvaja, zato ima vse tehnološke faktorje razen prvega enake 0. Prvi faktor je absorbirajoč, ker mora vsota vsake vrstice biti enaka 1 in ker so vsi drugi členi nič, mora biti prvi 1. Stolpci matrike nam kažejo tržni odnos, vrednosti členov matrike pa tehnološki odnos. Takoj lahko ugotovimo, da imata podjetji 1 in 3 enak tržni odnos, eno od drugega kupujeta. Zato ju lahko združimo. Faktorji podjetij 4 in 5 so ergodični, neodvisni od drugih, zato podjetji združimo.

Če je v presečišču vrstice i in stolpca j neka vrednost, to pomeni da podjetje j proizvaja za podjetje i, tako da podjetje j koristi podjetju i. Po tem principu narišemo diagram stanj sistema. Pričnemo s podjetjem P(0), ki ne proizvaja, pač pa upravlja s podjetji. Podjetje 1 je profitno in posredno preko njega tudi podjetje 3. Podjetje 2 je prav tako profitno in koristno podjetju 1 in 3. Koristno podjetju 2 je tudi podjetje 6, vendar to podpira podjetji 4 in 5, ki pa nista profitni. Zato se združijo samo podjetja 1,2 in 3 preko krovnega podjetja, druga pa se v tej združitvi opustijo. Smer puščic določa tehnološko pogojen pretok denarja.

$$P(0) \leftarrow P(1),P(3) \leftarrow P(2) \leftarrow P(6) \rightarrow P(4),P(5)$$

Nova, tehnološko tržna matrika je potem:

$$P = \begin{array}{c} \begin{matrix} P(0) & P(1) & P(2) & P(3) \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/4 & 1/2 & 0 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 1/2 & 0 & 1/2 \end{bmatrix} \end{array} \begin{matrix} P(0) \\ P(1) \\ P(2) \\ P(3) \end{matrix}$$

Stanje krovnega podjetja je absorbirajoče in iz matrike Q dobimo novo matriko N = 1/(I - Q), kjer je I enotina matrika. Matrika N določa proizvodnjo v stolpcu, ki jo stimulira enotino naročilo v vrstici. Tako nam proizvodni faktor n(i,j) določa vrednost proizvodnje podjetja j, ki ga stimulira enota vrednosti (1$) naročil na podjetje i. Matrika V sumira vrednost proizvodnje po podjetjih (vrsticah i):

$$N = \begin{bmatrix} 4 & 0 & 2 \\ 8/3 & 4/3 & 2 \\ 4 & 0 & 4 \end{bmatrix} \qquad V = \begin{bmatrix} 6 \\ 6 \\ 8 \end{bmatrix}$$

Tako nam naročilo 1$ na podjetje 2 stimulira proizvodnjo 8/3$ od P(1), 4/3$ od P(2) in 2$ od P(3). Na to naročilo napravi P(1) profit 8/3 x 1/4 = 2/3, P(2) profit 4/3 x 1/4 = 1/3 in P(3) profit 2 x 0 = 0, ker je P(3) neprofitno, skupaj tako podjetje 2 napravi 1$ profita.

Trg nastopa na začetku, ker z analizo trga ugotovimo njegovo kvaliteto in na koncu, ko določa kvantiteto tehnološkega procesa.

Na zahteve trga se podjetja odzivajo preko vrstičnega vektorja T, ki pove količino dolarskih naročil za podjetja, n.pr.:

$$T = (1,3,2)$$

Na tako zahtevo trga dobimo s produktom T x N = (20,4,16) proizvodnjo, ki jo ustvarijo posamezna podjetja. Totalna proizvodnja je določena s produktom T x V = 40$. V tej proizvodnji napravi P(2) 3$ profita. Podobno izračunamo profit še za drugi dve podjetji.

Dejstvo, da je podjetje lahko koristno tudi, če je neprofitno, potrjuje začetno tezo, da mora podjetje v tehnološko tržnem razmerju biti ali profitno, ali pa mora slediti profitnemu podjetju. Celoten proces ima po mojem mnenju izvor v inženirskem delu, ker sam trg in proizvodnja brez izumiteljskega in tehnološkega deleža ne zadoščata za profit podjetij.

Literatura: J.G.Kemeny, J.L.Snell, "Finite Markov Chains", Springer Verlag, N.Y., Heidelberg, Berlin, 1976.

Rihard Piskar

Na rob konferenci o zanesljivosti in kvaliteti

Na mednarodni konferenci, ki jo organizira IASTED (International Association of Science and Technology for Development) v Luganu, Švica, od 20. do 22. 6. 1989, je bilo med 72 prispevki svetovnih ekspertov v treh dneh mogoče slišati in videti tudi dva referata Iskre Delte, enega v sodelovanju s Fakulteto za elektrotehniko in računalništvo iz Ljubljane.

Prvi je bil "Zanesljivostna napoved ahitekture hiperkocke PARSYS" in drugi "Razpoložljivostno analitično orodje za kompleksne sisteme", ki sta poleg prodorne tematike paralelnega procesiranja in zanesljivostnega inženirstva predstavila širši javnosti del razvojnih dosežkov Iskre Delte.

Sodoben razvoj računalništva v svetu gre v dve smeri: superračunalnikov in paralelnih računalnikov. Prvi imajo poudarek na tehnologiji zelo visoke stopnje integracije in distribuiranem procesiranju, ki je kvaziparalelno, drugi pa poleg visoke tehnologije temeljijo na čistem paralelizmu.

Bistvo paralelizma je v tem, da ni več sukcesivnega izvajanja nalog, pač pa se vsaka naloga razbije na množico podnalog in se vsaka od njih istočasno in sinhrono izvaja. S tem se pridobi na hitrosti, ki je pomembna predvsem pri reševanju problemov, ki zahtevajo veliko obdelav v kratkem času, poleg tega pa na napakovni tolerantnosti, ki dopušča, da kak procesor ali del pomnilnika celo odpove, vendar pa drug procesor prevzame njegovo nalogo. Tak sistem je PARSYS, paralelni sistem Iskre Delte v raziskovalno razvojni fazi.

Posebno živahna diskusija se je razvila po predstavitvi analitičnega večrazsežnega sistema super-kub, za numerične evalvacije razpoložljivosti hierarhičnih, distribuiranih in paralelnih sistemov, ki deluje na sistemih Delta VMS in ga Delta s pridom uporablja. Bazira na večrazsežnostnih naključnih sistemih, ki prostorsko ponazarjajo stanja sistema, ki ga analiziramo in jih ovrednoti numerično, kar je nadvse pomembno pri prognozi obnašanja sistema v spreminjajočem se okolju. Sistem je interaktiven, trenutno je v fazi prenosa na sisteme Delta PC.

Sodeč po ugotovitvah s konference zanesljivost in kvaliteta nista sinonima. Zanesljivost, kot širši pojem, je često nepravilno uporabljen v konceptu kvalitete. Lahko se diskutira o tem, da je kvaliteta statični in zanesljivost dinamični deskriptor nekega procesa ali izdelka. Izložbeni izdelek je lahko visoko kvaliteten, toda njegova zanesljivost je izmerljiva samo z rabo, ali pa je ocenljiva s prognozo. Gledano na drug način je njegova kvaliteta intrinsična ali inherentna na izdelek sam, medtem ko je zanesljivost merilo njegovih performans v sistemu in določenem okolju.

Zanesljivostni pristop k problemu zadovoljstva kupca je posledica kombiniranega razvoja kvalitete in zanesljivosti. Znano je, da so Japonci pred 15 leti dobivali instrukcije o kontroli kvalitete iz ZDA. Dobili so model in uporabljali so ga. Toda ne samo, da so ga uporabljali, tudi izboljšali so ga. Večina vseh metod kontrole kvalitete od motivacij ljudi, zero defektov, treniranju osebja iz principov kontrole kvalitete se je pričelo v ZDA. Danes se v ZDA že učijo od Japoncev in se zavedajo bazične pomembnosti zanesljivosti in kvalitete za podjetja in njihov vpliv na tržišče in poslovno strategijo v področju razvoja, strokovni spretnosti in pogodbenem odnosu s kupci.

Na drugi strani malo Japoncev spodbija vodilen vpliv ameriških inženirjev na področju zanesljivostnega inženirstva, prav tako kot priznavajo ameriško vodstvo v sodobnem razvoju in tehnologiji. Čeprav Japonci izvrstno upravljajo proizvodnjo in marketing visoke kvalitete predvsem za potrošniške izdelke, je tehnologija za temi produkti dobro znana. Prav tako je zelo malo japonskih projektov, ki zahtevajo visoko zanesljivost kot jo zahteva vesoljski raziskovalni program. Japonski cilj je osvojiti zanesljivostno teorijo, razvoj in inženirstvo. Medtem pa je jasno, da zna predstavljati tradicija Japoncev, ki zavira razvoj individualnih dosežkov in spoznanj v korist timskega dela, zanje težavo, da bi se istovetili z njimi nasprotno ameriško tehnološko kreativnostjo.

Na vprašanje, kakšen model razvoja zanesljivosti in kvalitete uporabiti v Jugoslaviji, menim, da nikakor ni priporočljivo, da spoznamo vse faze razvoja, kot ZDA ali Japonska, pač pa se lahko na razvoju teh dveh naučimo in nadaljujemo pri njunih izkušnjah. Jugoslovanska svobodnost mišljenja in kreativnost, ki je bližja ameriškemu načinu, nam daje misliti, da nasilno usmerjanje v japonski model ne bo prineslo zaželjeni učinek. Zato je predvsem v sorazmerno začetni fazi razvoja zanesljivostnega inženirstva in kvalitete zelo pomembno sprostiti kreativni potencial, ki ga imamo in ne podrediti individualnih dosežkov skupnim

ciljem. Vkolikor bi že na začetku določili okvire, v katerih se lahko razvija zanesljivostno inženirstvo kot del aktivnosti kontrole kvalitete, kot kažejo težnje v nekaterih okoljih, bi hkrati zadušili kreativno mišljenje, brez katerega razvoj ni mogoč. Tudi na pričujoči konferenci je bilo 6 referatov jugoslovanskih, ki dokazujejo jugoslovansko kreativnost v razvoju konceptov zanesljivosti in kvalitete sodobnih tehnoloških dosežkov.

Rihard Piskar

Informacijska tehnologija v Španiji

Špansko gospodarstvo je v razcvetu in celotno področje informacijske tehnologije raste nezadržno z ekonomskim napredovanjem podjetij. Čeprav na španskem trgu podjetje IBM še vedno usmerja tržne potrebe pa vrsta tujih proizvajalcev sodeluje v španskem razcvetu.

Španija je dolgo samevala na slepem koncu Evrope, odkar so njeni konkvistadorji osvojili Latinsko Ameriko. Naenkrat pa je v zadnjih nekaj letih postala gospodarsko najprivlačnejša dežela. Njena brsteča ekonomija je uravnotežena med prednosti podrazvitosti - cenenim delom in neizkoriščenim kapitalom - in prednostmi, ki izvirajo iz članstva v Evropski skupnosti.

Rastoča španska ekonomija vleče s seboj tudi informacijsko tehnologijo; obe sta posledica naraščajočega bogastva in sposobnostne tehnologije, ki omogoča ekonomsko ekspanzijo. V zadnjih petih letih je računalniška prodaja v Španiji narasla za 321%, in sicer prek 4 milijarde dolarjev. Proizvodnja računalniške industrije se je povečala za 335%, in sicer na približno 1 milijardo dolarjev.

Prodaja računalniške aparaturne opreme je pokrila 60% trga, vzdrževanje, šolanje in svetovanje približno 21%, programska oprema pa približno 17% trga. Vsa tri področja izkazujejo letno povečevanje dohodka med 20% in 30%. Še posebej so narastle potrebe po osebnih računalnikih. Leta 1985 je le 2,5% poslovodnih delavcev uporabljalo osebne računalnike. V letu 1987 jih je bilo že 6%, v letu 1992 pa naj bi dosegli razmerje 32%.

V tem deviškem prodajnem okolju ni več presenetljivo, da je Španija postala privlačna za tuje investitorje. Manj kot 7% španskega industrijskega dohodka izvira iz čistih domačih (domorodnih) podjetij. Od leta 1983 se je uvoz povečal za 327%, in sicer na vrednost 3,404 milijarde dolarjev - štiri od petih prodanih osebnih računalnikov v Španiji so uvoženi.

V Španiji je kar nekaj multinacionalk, ki nimajo svojih podružnic ali distribucijskih zastopnikov. Podjetja za proizvodnjo in sestavljanje v Španiji so npr. Hewlett-Packard, ki izgrajuje v svojem glavnem središču za periferijo novo tovarno, razširja pa tudi svojo tovarno v Barceloni; tudi proizvodnja tiskalnikov in avtomatičnih blagajn podjetja Fujitsu se je povečala za 35% in 75% v zadnjem letu.

Podjetje Fujitsu-Espana proizvaja v svoji tovarni v Malagi tudi miniračunalnike, mikroračunalnike in naprave za pisarniško avtomatizacijo; v to tovarno je podjetje investiralo v zadnjem letu nadaljnjih 10,5 milijonov dolarjev. Podjetje Fujitsu ima svoj glavni evropski urad v Madridu in njegova španska podružnica Fujitsu-Espana je zelo uspešna. Podjetje se je usidralo v Španiji pred 15 leti in je začelo skupni posel s španskim računalniškim proizvajalcem Secoinsa, in sicer z proizvodnjo v Španiji konstruiranega računalnika. V letu 1986 je španska PTT Telefonica odkupila 40% delež. Podjetje Fujitsu-Espana je pridelalo v razdobju 1987/88 dobiček 7,6 milijona dolarjev pri dohodku 160 milijonov dolarjev, kar je pomenilo povečanje dobička za 27%; dobiček naj bi se v tem letu povečal še za 50%.

Kljub uspešnosti Fujitsu-Espana v Španiji pa je podjetje IBM še vedno prevladujoče na španskem računalniškem trgu. Na IBM odpade kar 36% celotne industrije, in sicer kot na dobavitelja računalnikov in izvoznika. Njemu sledita Philips kot proizvajalec terminalov in Epson kot prizvajalec tiskalnikov. Druge multinacionalke s pomembnimi deleži v Španiji so še Unisys, Olivetti, Nixdorf, Xerox, DEC in NCR. Na eksplozivnem trgu osebnih računalnikov je značilno prodrlo podjetje Amstrad z modeloma 1512 in 1640 in podjetje Altos, ki je izdobavilo 5000 večuporabniških mikroračunalnikov v zadnjih šestih letih. Na veliko večjem francoskem trgu je Altos prodal samo 10000 sistemov v 10 letih. Altos napoveduje, da bo po letu 1992 naraščal španski trg za večuporabniške sisteme hitreje kot v katerikoli drugi evropski državi.

Vstop podjetja ICL na špansko prizorišče sega 10 let nazaj, ko je bilo odkupljeno Singerjevo mednarodno računalniško podjetje na podedovanem španskem tržišču. Konec lanskega leta pa je ICL oblikoval skupni posel (joint venture) z andaluzijskim razvojnim institutom, da bi razvijal programsko opremo in storitve, namenjene zdravstvenim, izobraževalnim, storitvenim in javnim ustanovam. Drugi, vertikalni tržni interes ICLa v Španiji je trgovina na drobno. Skupni posel podjetja ICL-Sistemas obvladuje iz Sevilje le 30 inženirjev in tehnikov. ICL vzpostavlja tudi mednarodni nakupni urad v Barceloni, ki bo preizkušal španske izdelke glede na njihovo ustreznost za izvoz v ICLove tovarne kjerkoli. ICL bo oblikoval tudi reprezentativno središče v Madridu, kjer bo razvijal in prodajal aparaturno in programsko opremo po vertikali španskim vladnim ustanovam.

Španija postaja eno najpomembnejših razvijajočih se področij Evrope. Tako se razvijajo tudi številne računalniške svetovalne organizacije v Madridu, Barceloni in Sevilji, skupni izobraževalni center pa je Segoviji. Čeprav tudi na španskem trgu dominirajo multinacionalke, kažejo španska podjetja za informacijsko tehnologijo veliko zagnanost in ustvarjalnost. Investronica je španski pionir na področju osebnih računalnikov in obvladuje 30% tržišča z 250000 instaliranimi enotami. To podjetje je bilo ustanovljeno pred devetimi leti kot tehnični oddelek tekstilne industrije Induyco, ki je sedaj član grupacije El Corte Ingles. Tako je Investronica lahko izkoristila aplikativne niše v računalniško podprtem oblikovanju in proizvodnji tekstilne industrije, ko je izvažala sisteme z delovno postajo v 30 držav.

Vodilno špansko podjetje za programsko opremo je Centro de Calculo de Sabadell (CCS), ki je nastalo pred 25 leti, vendar je v letu 1972 postalo vodilno špansko podjetje za razvoj teleprocesorske mreže. 75% podjetja je v francoski lasti, sodelovalo pa je v različnih raziskovalnih projektih v Evropi, vključno v projektu Esprit. Podjetje prodaja npr. Masterpat, ki je programski paket za upravljanje patoloških primerkov in npr. tudi pogodbo za računalnik in programsko opremo za elektronsko in telekomunikacijsko metodologijo za olimpijske igre leta 1992 v Barceloni. Dohodek tega podjetja v letu 1988 je znašal $46m, kar je 20% več kot leto poprej.

Tudi podjetja na spodnjem koncu tržišča iščejo priložnosti za svojo razširitev. Npr. podjetje Telmatica e Informatica Internacional je razvilo izboljšano storitev za videotex, ki je dosegljiva prek centrov v več evropskih državah. Drugo majhno podjetje Biochip je izkoristilo množična potovanja po Španiji s prodajo aparaturne in programske opreme za hotelske rezervacijske sisteme.

Tudi vlada podpira špansko računalniško

industrijo, ko vzpodbuja sodelovanje s tujimi podjetji, toda vpliva na tiste multinacionalke, ki smatrajo, da je Španija le dežela sestavljavcev in izpuh za prodajo; pri tem zagovarja zlasti dejavnosti s področja raziskav in razvoja. Siemens, Olivetti in Unisys imajo v Španiji svoja raziskovalno-razvojna središča. IIP, Olivetti in Bull so se vselili v Valeški tehnološki park v Barceloni, ki je le eden od štirih tehnoloških parkov, v katere je vlada investirala $55m. Fujitsu-Espana ima tri raziskovalno-razvojne centre v Madridu, Malagi in Barceloni, kjer je bila tudi razvita vmesniška plošča za PCjevsko delovno postajo, ki se izvaža v Korejo.

Ta gibanja spremlja ministrstvo za industrijo, ki ugotavlja, da se je v Španiji prvič zgodilo to, da pri njih razvijajo nove produkte tudi multinacionalke. To pa je morda veliko pomembnejše od vrste tujih tovarn za sestavljanje. Za vse majhne države so domače raziskave in razvoj zelo drago podjetje. Pri vzpodbujanju sodelovanja s tujci participira Španija tudi v novi program Esprit. Od 155 predloženih španskih projektov jih je bilo sprejetih 74. Ti projekti se tičejo 50 španskih podjetij in 10 univerz. Investronica dela na projektu za razvoj produkcijskega sistema za tekstilno industrijo in podjetje Ibermatica prispeva $750k k $21m, kolikor je predvidenih za projekt, s katerim se razvija generični vizijski sistem za industrijsko uporabo. CCS sodeluje v projektu za vzdrževanje programske opreme in nacionalno telekomunikacijsko podjetje Telfonica v 10 komunikacijskih projektih.

V projektih Esprit sodelujejo španske podružnice multinacionalk, npr. Siemens in Katalonska univerza, ki izdelujeta prevajalnik za podatkovne baze; Alcatel Standard Electrica razvija sistem za računalniško podporo za primere katasrofe. Španska vlada je začela uresničevati tudi drugo fazo svojega Nacionalnega elektronskega in računalniškega projekta (PEIN II), ki se začel v letu 1984. Strategija vlade je, da oblikuje večje industrijske grupacije, še posebej dve združevalni krovni podjetji (holdinga) za področje telekomunikacij in programske opreme, s ciljem da bi okrepila špansko konkurenčnost do leta 1992.

Vladna ministrstva se pooblaščena, da sodelujejo v investicijah v velike projekte informacijske tehnologije, ki naj bi povezovali domorodna in multinacionalna podjetja. Iris je npr. ime projekta v vrednosti $21m, v okviru katerega se razvijajo teleinformacijske storitve in homogena komunikacijska mreža za španski sektor raziskav in razvoja. SCTM je vojaški komunikacijski projekt, Bubi pa je projekt za avtomatizacijo velikih tobačnega naročniškega sistema z vrednostjo $30m. PEIN II je investiral $150m v elektronske in računalniške projekte že v letu 1984 in $480m v letu 1986. Zneska $136m in $144m sta predvidena za investiranje v tem in prihodnjem letu.

Španska informacijska industrija pa ne oblikuje le zdrav domači trg, temveč vztrajno povečuje tudi izvoz. Špansko združenje za izvoz elektronike in računalnikov stalno propagira in vzpodbuja izvoz informacijske tehnologije. Investronica je povečala izvoz za 94% v zadnjih petih letih ($30m). Podjetje ne izvaža le v Evropo temveč tudi v Južno Korejo, Japonsko in Argentino. Kljub vsej tej aktivnosti pa je španska informacijska industrija še precej oddaljena od možnosti, da se v letu 1992 pojavi v Evropski skupnosti z vso svojo močjo.

Španija mora še preseči svoj lasten odpor do računalniške uporabe. Videotex je doživel v Španiji popoln neuspeh, saj latinski temperament bolj ceni osebni kontakt. Splošno obotavljanje v uporabi računalnikov je značilnost, katere posledica je npr., da je bilo v letu 1987 v uporabi manj kot 6000 računalniških blagajn in le s 3,4 transakcijami na osebo; v Združenem kraljestvu je znašalo to razmerje 8 transakcij na osebo. Povprečna letna prodaja na kreditne kartice doseže le $85 na osebo. Vse to kaže na določeno informacijsko nespretnost španske populacije in je problem, ki je računalniški industriji znan širom po svetu.

KOMENTAR. Španski primer je prav gotovo lahko zanimiv za jugoslovansko in slovensko politiko in podjetništvo na področju informacijske tehnologije. Kaj so naredile vse naše vlade in vladne institucije na področju vzpodbujanja, investiranja in organizacije računalniške industrije? To kar je mogoče z vso zanesljivostjo trditi, je, da niso storile ničesar pomembnega, integrativnega, diplomatskega in nazadnje tudi industrijsko naprednega. Domača računalniška industrija nam je obtičala v spontanem razsulu. Celo napori, da se oblikuje razvojni diskurz med politiko in industrijo, so ostali neuresničeni zaradi pomanjkanja političnega pa tudi podjetniškega interesa. Gospodarska politika se nam je zreducirala na načelo, da naj se vsak rešuje sam. Ali smo posamezniki sploh poklicani in kvalificirani, da se ubadamo s problemi gospodarske in tehnološke politike? Zakaj potem sploh imamo vladne in družbene institucije, ki bdijo nad možnostmi in nujnostmi razvoja?

Ko se sprašujem, kako bi lahko vzpodbudno napisal poročilo z naslovom "Informacijska tehnologija v Sloveniji in Jugoslaviji", ki bi bilo primerljivo s Španijo, moram vendarle povedati nekaj več resnice. Španski čudež je seveda ekonomski in šele v okviru tega je mogoče pričakovati spodbudne rezultate v razvoju informatike kljub španskemu temperamentu in zavračanju informatike kot sodobnega pojava človeštva. Na nek način je celo uspešen španski razvoj vsaj na področju informatike konzervativen, pogojen z mentaliteto populacije. Pri nas je verjetno ta mentalna ovira spektralno širša in se ne omejuje le na informatiko, prežema gospodarstvo, podjetništvo in politiko. In dokler je tako, bo informacijski zaostanek le posledica ekonomskega in političnega. In tu bržkone lahko presahne tudi smiselnost vsakršnje zasebne pobude.

A. P. Železnikar

DECove evropske delovne postaje prihajajo iz podjetja Olivetti

Digital Equipment (DEC) in Olivetti sta se sporazumela, da bo Olivetti dobavljal DECove osebne računalnike za evropsko tržišče. Osebni računalniki bodo proizvajani po DECovih specifikacijah v Olivettijevih tovarnah in trženi in vzdrževani prek DECovih poslovnih in servisnih organizacij. V okvir tega programa spadajo osebni računalniki s procesorji 80286, 80386SX in 80386, ki oblikujejo podlago za DECove nove osebne delovne postaje. DEC se je odločil za italjanskega proizvajalca, ker je Olivetti močno prisoten na evropskem trgu.

A. P. Železnikar

Vrednost DECovih akcij je padla

Vrednost DECovih akcij je padla za dobrih 10%, ko je postalo znano, da bo dobiček v tretjem poslovnem kvartalu (v marcu 1989) manjši od

pričakovanega. Od pričakovanoga porasta dobička 13% do 31. marca je bil dosežen prirastek le 10%. Kot utemeljitev za ta "neuspeh" se navaja poslabšanje na ameriškem trgu in problemi z novimi produkti.

A. P. Železnikar

Ali se bodo načrtovalci hardwara morali naučiti programiranja?

Prihaja bržkone čas, ko se bodo ožičevalni inženirji morali sprijazniti s kodiranjem (programiranjem). Verjetno ste že slišali izrek: hardware je software. (Tudi sam pravim, da je arhitektura ali struktura stroja ali hardware informacija.) Ta čas prihaja, saj načrtovanje hardwara postaja vaja iz kodiranja. Inženirsko delo je v primeru integriranih vezij le še pisanje koda, ki je čedalje bolj podoben programu v jeziku C ali Adi.

Ko je naraščala popularnost računalniško podprtega načrtovanja, ni bilo težko predvidevati, da bo z računalniki mogoče kmalu sestavljati tabele in ne več risati diagrame (načrte povezanih enot), da bo mogoče definirati module, načrtovalne procedure itd. s programskimi vrsticami. Za te namene je tako nastal jezikovni standard VHDL. Ta okrajšava pomeni: "V" kot VHSIC za pentagonski program Very High Speed Integrated Circuits; HDL pomeni Hardware Description Language. V zadnjem letu je bila v okviru IEEE izdelana verzija jezika, ki ga podpira obrambno ministrstvo ZDA, v igro pa se je vključilo tudi podjetje Mentor Graphics Corp., San Jose, Kalifornija.

Prihaja čas, ko se bodo hardwarski inženirji morali začeti učiti pisanja koda v jeziku VHDL. Trg t.i. VHDL-simulatorja je še majhen. Mnogi opazovalci dvomijo, da bodo hardwarski inženirji zmogli radikalni obrat v svojem načinu razmišljanja. Zaradi tega je npr. podjetje Vantage Analysis Systems Inc. skrilo sam jezik pred uporabnikom in s simulacijo prikazalo znane shematične grafe s knjižnjico funkcijskih modelov v VHDL. Ta način načrtovanja pa je brezno, ki ga veliko inženirjev ne bo zmoglo preskočiti.

A. P. Železnikar

DEC napada pisarniško tržišče

Digital Equipment je vrgel na tržišče dva računalnika tipa Microvax, s katerima naj bi resno napadel tržišče pisarniške opreme, na katerem gospodujeta sedaj podjetji IBM in Wang. Microvax 3800 in 3900 naj bi zamenjala modela 3500 in 3600 s 50% izboljšanim razmerjem cena/zmogljivost in s štirikratno povečanim obsegom pomnilnika. DEC tudi počasi upokojuje stari in še vedno prodajani Microvax II, ki ga je vrgel na tržišče v letu 1985, ko je znižal ceno aparaturne in programske opreme glede na prejšnje svoje modele. DEC izjavlja, da bo v prvem naletu naskočil tržišče večuporabniških sistemov z 8 do 40 uporabniki. DEC uvaja tudi skupinski (clustered) sistem Microvax in ponuja skupinski Vax.

Dva Microvaxa 3800, povezana z DSSI (Digital Storage System Interconnect), bosta dopuščala katerikoli uporabniški dostop do kateregakoli pomnilnega elementa, ki bo privezan na enega od gostiteljev. Očitno uporablja DEC podobno taktiko, kot jo je uporabil pri zamenjevanju sistemov 6200 s sistemi 6300, ko ponuja dopolnilne pakete pri samo 5% povečanju cene.

A. P. Železnikar

Sistemska integracija

Denar se skriva v t.i. orkestraciji. Padajoči profiti prepričujejo tudi največje računalniške proizvajalce o nujnosti integracije. IBM izjavlja, da bo prodajal Vaxe, če bo dobil dovolj veliko naročilo. En sam računalniški proizvajalec dejansko ne more biti neodvisni sistemski integrator. Letna poročila podjetij so poučna. In v zadnjem času postajajo zgodbe o proizvajalcih aparaturne opreme čedalje bolj žalostne.

Trg miniračunalniških izdelkov se bo v letu 1989 bržkone skrčil. Podjetje Control Data je v letu 1988 pridelalo le skromnih $1,7m dobička, norveški Norsk Data pa že izgubo $129m. Tudi podatki o poslovanju IBM iz zadnjega obdobja kažejo, da se pridobiva dohodek na področjih, ki niso hardwarska. V letu 1987 je plasma softwara prinesel 13% od celotnega prihodka, medtem ko je znašal v letu 1983 le borih 6%. Sporočilo je glasno in jasno: za posel je potrebno imeti kaj več kot le železnino.

Zaradi tega ni presenetljivo, da se računalniški proizvajalci pomikajo na področje sistemske integracije. Nič več se ne zadovoljujejo z instalacijo novih mainframov (glavnih računalnikov) s prepuščanjem ostalih postavk v projektu konkurenci ali specialistom s področja računalniških mrež. Vse bolj se zavzemajo za vlogo glavnega kontraktorja (pogodbenika) in za kontrolo nad profitom, ki je s tem povezan. V zvezi s tem izjavlja npr. IBM, da bo prodajal DECove Vaxe, če mu bo to pomagalo, da si pridobi posel naročnika. S tem pa se bistveno spreminjajo nekdanje opredelitve velikanov, ki bi v ne tako davnih časih veljale za nesprejemljivo poslovno in tehnološko krivoverstvo.

Podjetje Computing Services Association (CSA) navaja, da je bilo v letu 1987 tržišče softwara in storitev v zapadni Evropi vredno $32bn in četrtino tega dohodka so poželi proizvajalci hardwara. V tej areni se pojavljajo tradicionalni računalniški proizvajalci vključno z IBM, DEC, Unisys in Hewlett-Packard in z njimi se bo ta delež še povečeval. Toda kateri tržni segment imajo vsi ti proizvajalci v mislih? Ideja o sistemski integraciji je po mnenju CSA predvsem "modna zamisel".

Seveda so dobavitelji že od nekdaj poskušali povezovati svoje produkte z napravami drugih proizvajalcev z uporabo mrež in raznovrstnimi drugimi storitvami, če se je to pokazalo kot potrebno. Razlika pa je zdaj v tem, da je bila ta praksa enkapsulirana v termin "sistemska integracija". Seveda pa se naraščanje sistemske integracije kaže kot dramatično, če se upošteva, da se je zgodila evolucija v smeri koncepta

podjetniške in tehnološke koeksistence. Nekateri sicer zatrjujejo, da je termin sistemska integracija znan že 25 let, vendar gre v zadnjem primeru tudi za bistvene terminološke spremembe. Definicija je lahko tudi tale: sposobnost integracije različnih aparaturnih in programskih okolij v enotno poslovno rešitev. To pa je mogoče razumeti in imenovati kot posebno storitev. In treba je priznati, da takšno storitev lahko nudi le dovolj usposobljen in poslovno prožen proizvajalec.

Nova paradigma sistemske integracije se je razvila pri uporabnikih. Uporabnik si enostavno želi več storitev. Računalniški centri ostajajo neizkoriščeni zaradi stalnega manka sposobnih kadrov in uporabniki iščejo pomoč tako pri izbiri sistema kot pri njegovi uporabi pri poslovanju. Sistemski integrator mora zato bolj razumeti potrebe uporabnikovih poslov kot pa kaj drugega. To pa je lahko prednost le ustrezno izkušenih integratorjev, ki se ji navadni prodajalci seveda ne morejo približati. Proizvajalci lahko take integratorje najamejo in jih posebej plačajo, medtem pa intenzivno izobražujejo svoje prodajalce.

Nič čudnega torej ni, če je kontrolni položaj v obsežnem in zahtevnem projektu visoko cenjen. Zato se večina računalniških podjetij profesionalizira za vlogo t.i. glavnega kontraktorja v poslih sistemske integracije. Toda tudi večina bi požrla slavo, če bi le lahko bila udeležena pri delitvi dobička. IBM npr. izjavlja, da bi lahko prodal karkoli od zahtevanega, če bi bili posamezni deli sistema dopolnjeni z DECovimi Vaxi ali z Amdahlovim mainframom. Pri tem praktično ni omejitve v tem, kdo vse bi lahko bil subkontraktor. Edina omejitev je poslovno tveganje. Teoretično je sprejemljivo, da se npr. nekaj kupi od DECa in potem preproda. Toda bolj realističen scenarij bi bil povezava novega IBMovega sistema v obstoječo DECovo bazo. Obratno pa bi bilo mogoče tudi k IBMovi bazi dodajati DECove naprave.

Zaradi določenih pojavov omahljivosti pri računalniških proizvajalcih, ko gre za sistemsko integracijo, se odpirajo možnosti prav t.i. tretjim integratorjem. Tako se lahko posebna integratorska podjetja ustanavljajo in vzdržujejo za različna področja uporabe s pomočjo računalniških proizvajalcev. Potrebni pa so tudi neodvisni konzultanti, ki jih plačujejo uporabniki.

Jasno je pri vsem tem, da se bo področje sistemske integracije moralo še razvijati, preden bo doseglo svoje logične meje, npr. pri gradnji avtomatizirane trgovine in spremljajoče tehnologije v njej. Medtem pa ostaja ključen problem medsebojna povezava podsistemov. Napori podjetja Unisys v smeri odprtih sistemov bi lahko to nalogo olajšali: teoretično bi lahko njegova t.i. Povezava odprtega sistema komunicirala s katerimikoli vstavljenimi stroji. Vendar standarizacija ne bo nikoli dosegla tiste ravnine, na kateri bi hardware postal nepomemben. Tudi če bi Unix postal zares prenosljiv, bodo proizvajalci še vedno lahko inovirali svoje stroje.

Glavno načelo sistemske integracije pa ostaja prej ko slej tole: ključna lastnost integracije je, da sistemi delujejo skladno in dovolj učinkovito. To pa naj bi popolnoma osrečilo tudi uporabnike.

A. P. Železnikar

Računalništvo kot postavje matematike

Matematiki so pomagali pri oblikovanju računalniške znanosti, ta pa sedaj povzroča revolucijo svojega lastnega subjekta. Kakšen je ta dramatičen doprinos?

Vrsta matematikov se je vpisala v zgodovino računalništva. Charles Babbage je konstruiral predhodnika sodobnega digitalnega računalnika že v letu 1830. Alan Turing je definiral svoj abstraktni stroj že v letu 1930 in vplival na razvoj teorije izračunljivosti. John von Neuman je razvil zamisel sekvenčne arhitekture.

Medtem je razvoj računalništva že turbulentno vplival na samo matematiko. S prebijanjem skozi suhoparna števila enačb, preiskovanjem milijonov možnosti in z uporabo grafike je ta razvoj povzročil nove koncepte in približal računalnike matematikom kot njihovo bistveno orodje. Kljub temu pa vrsta matematikov verjame, da računalništvo kompromitira in uničuje fundamentalne vrednote prave matematike.

Npr. v letu 1976 sta Keneth Appel in Wolfgang Haken dokazala notorični teorem štirih barv. Ta teorem pravi, da je s štirimi barvami mogoče obarvati karterikoli zemljevid tako, da dve sosednji območji nista enake barve. Od leta 1852 dalje so znani matematiki zaman poskušali dokazati ta izrek. Zato je izjava o pravilnem dokazu tega izreka povzročila veliko vznemirjenje. Za vrsto matematikov je bila ta zmaga jalova: dokaz je zahteval uporabo računalnika.

IBMov računalnik 360 je odigral ključno vlogo z izračunavanjem tisočev primerov in je pustil matematike v dvomu, kaj bi v bistvu lahko konstituiralo dejanski dokaz. Nobeno človeško bitje ne more verificirati izračunov, toda večina matematikov sprejema te izračune kot veljavne, čeprav je potrebno zaupanje, da računalnik ni naredil napake. Kljub tem težavam, ki se tičejo fundamentov matematičnega subjekta, so ostali matematiki 20. stoletja izredno ustvarjalni.

V svoji knjigi "The creative computer" pravita Donald Michie in Rory Johnson: "Že dolgo se napačno predpostavlja, da je iz računalnika mogoče dobiti le tisto, kar je bilo vanj vstavljeno. ... Sedaj pa je bilo neizpodbitno pokazano, da lahko pride iz računalnika tudi nekaj novega in to novo je znanje. To znanje pa so lahko tudi izvirne ideje, strategije in rešitve realnih problemov."

Na prvi pogled izgleda absurdno, da je to res prav v matematiki. Ali je matematika zgolj neka vrsta računalništva oziroma računalniške rutine? Računalništvo kreira novo obliko eksperimentalne matematike, generira nove matematične strukture in probleme in omogoča nove poti modeliranja zapletenih sistemov z uporabo podatkovnih baz in izračunov. Fraktalna geometrija je le eno od dobro znanih vej nove matematike, ki je omogočena z računalništvom. Njeni antecendenti vključujejo snežinke, ki jih je opisal švedski matematik Helge von Koch leta 1904. Te snežinke je mogoče oblikovati iz enostavnega trikotnika s ponavljanjem "samopodobnih" operacij na robovih. Čeprav je mogoče matematično dokazati, da ima meja snežinke neskončno dolžino pa ima snežinka končno površino, ki je enaka očrtanemu krogu izvirnega trikotnika.

Benoit Mandelbrot, IBMov raziskovalec in oče fraktala, postavlja fractal v zgodovinsko perspektivo krize matematike, ki se je sprožila že z delom Giuseppa Peana. Leta 1890 je Peano definiral krivuljo, ki je zvezna in poteka skozi vsako točko kvadrata. Peanova krivulja omogoča specifikacijo točke z enim samim številom, z njeno oddaljenostjo od konca krivulje. Opredelitev dimenzije kot števila spremenljivk, ki so potrebne za specifikacijo točke, je postala nevzdržna. Kriza se je končala v letu 1922, ko je Besicovitch predložil končno obliko tega, kar se danes imenuje Hausdorff-Besicovitcheva ali fraktalna dimenzija. V primeru zvezne krivulje se dimenzija nanaša na frakcijo področja ravnine, ki jo pokriva.

Mandelbrot definira fraktalno množico kot tisto, ki ima Hausdorff-Besicovitchevo dimenzijo večjo od njene topološke dimenzije. Topološka dimenzija ustreza navadno intuitivni ideji dimenzije kogarkoli. Žica je enodimenzionalna, plošča dvodimenzionalna, bloki so tridimenzionalni itd. Krivulja Peano-Hilberta ima fractalno dimenzijo 2 in proti-intuitivno je njena topološka dimenzija 2. V letu 1979 je Mandelbrot preizkušal tole enostavno iterativno formulo: vzemi neko kompleksno število, ga kvadriraj, prištej k njemu konstanto in tako naprej. Ali bo končni rezultat divergiral v neskončnost, konvergiral k fiksnemu številu ali izkazoval neko vmesno stanje? Odgovor je, da ležijo konstante, ki povzročajo konvergentno stanje, v obliki hrošča, ki je znana kot Mandelbrotova množica. Vendar je zgodba še zanimivejša: če se Mandelbrotova množica izračunava in prikazuje na zaslonu, se pojavijo posamezni otoki, ločeni od telesa. Vendar matematični izreki pokažejo, da so otoki povezani s telesom množice in ta pojav imenuje Mandelbrot "vražji polimeri". Značilna lastnost teh množic je samo-podobnost. Če računalnik gleda v to množico s povečavo v posamezne njene dele, je mogoče opaziti, da so podstrukture podobne bolj grobim strukturam.

Ena najprivlačnejših vej matematike je trenutno teorija kaosa. Čeprav je ta teorija v tesnem razmerju s fraktali in Mandelbrotovo množico, je pojem kaosa odkril neodvisno od fraktalov Edward Lorentz v letu 1980. Bil je presenečen, ko je odkril, da že majhne spremembe začetnih vrednosti pri simulaciji vremena povzročajo znantno divergenco po preteku določenega časa. Te vrste pojavnost se imenuja kaotičnost. Če se predpostavlja, da so enačbe pravilne, to pomeni, da lahko že zelo majhne napake v opazovanju povzročijo dramatične posledice. Ta pojav je mogoče opisati kot metuljčni efekt: metulj, ki razpre svoja krila danes v Pekingu, lahko oblikuje viharni sistem drug mesec v New Yorku.

Morda zveni paradoksalno, da ima kaos strukturo. Množica točk, v kateri obstaja kaotična trajektorija, se imenuje nepričakovani atraktor: atraktor zaradi tega, ker sistem ne pobegne iz njega in nepričakovani, ker nekatere trajektorije niso periodične in zaradi načina, kako se trajektorije globoko prepletajo ena z drugo. Računalniško omogočena matematika fraktalov lahko najde pomembno področje uporabe v novi vrsti znanosti o modeliranju kompleksnih sistemov, ki so bili doslej deterministično utemeljeni.

Teorija avtomatov je nadaljnje področje matematičnega interesa, kot je pokazal John Conway v sijajni Igri življenja (Game of Life). Life je mogoče igrati obsesivno na PCju, ko se generacije barvnih konfiguracij razvijajo na zaslonu. V matematiki se je zgodil bistven premik v smeri, da bi postala eksperimentalna znanost. Obstajajo računalniški paketi, ki pomagajo matematikom pri vsakdanjem delu, podobno kot pomagajo procesorji tekstov pisateljem.

Potrebe računalniške znanosti so tudi vzpodbudno vplivale na razvoj novih področij matematike. Pisanje korektnih programov naj bi bilo podvrženo enaki matematični strogosti, kot je npr. oblikovanje tehničnih konstrukcij. Zanesljivost programov postaja čedalje bolj pomembna v industriji programske opreme. Novi specifikacijski jeziki, kot sta Z in VDM, so zelo blizu matematiki in so utemeljeni z matematičnimi teorijami. Narava metrike v programirni tehniki vzbuja pozornost matematikov, kot je npr. zamisel o programski strukturi.

Neglede na povedano pa začenja računalništvo vplivati tudi na poučevanje matematike. Interaktivna barvna grafika lahko znatno poživlja tradicionalno dolgočasna in suhoparna študijska področja, motivira in ilustrira zapletene ideje in tako zmanjšuje frustracije, ki se pojavljajo ob nepravilnih odgovorih pri trivijalnih napakah izračuna. Naslednje stoletje bo preživljala matematika srečneje s svojimi računalniki: ojačevanje matematične domiselnosti in možganskih zmogljivosti bo postalo naravno in dobrodošlo kot del vsakdanjika - tako kot sta bila včasih tabla in kreda.

Literatura:

[1] The Mathematical Revolution Inspired by Computing (Conference held at Brighton Polytechnic on April 5-7, 1989);
[2] D. Michie and R. Johnson, The Creative Computer, Pelican 1985;
[3] I. Stewart, The Problems of Mathematics, Oxford University Press 1987;
[4] K. Devlin, Mathematics: the New Golden Age, Penguin Books 1988;
[5] B. Mandelbrot, The Fractal Geometry of Nature, Freeman 1983;
[6] Peitgen and Saupe, The Science of Fractal Images, Springer 1988;
[7] J. Gleik, Chaos, Heinemann 1987;
[8] A. Holden, Chaos, Manchester University Press 1986;
[9] W. Poundston, The Recursive Universe, Oxford University Press 1987.

A. P. Železnikar

IBM je predstavil svoj prvi sistem s procesorjem 80486

IBM prevzema spet vodilno vlogo na področju PC tehnologije. Pred kratkim je predstavil novo tiskano ploščo, ki vsebuje Intelov procesor 80486. S tem je IBM prvi proizvajalec, ki ponuja produkt s procesorjem 80486. Tako v ZDA kot v Evropi se IBM spopada s podjetjem Compaq pri prodaji sistemov s procesorjem 80386. Compaq si je že priboril naskok kot tehnološki prvak na trgu s prvim sistemom 386, s taktno frekvenco 33 MHz, kar omogoča zmogljivost 10 MIPS.

S svojo vtično ploščo tipa 486 dovoljuje IBM njeno uporabo v PS/2 modelu 70, kjer doseže zmogljivost 15 MIPS. Ta zmogljivost pa je bila doslej prihranjena za superminije in spodnje modele mainframov. Nova plošča se nasadi na

glavno ploščo modela 70 in zamenja tako pomnilniški vmesni krmilnik 80385, procesor 80386 in koprocesor 80387 s procesorjem 80486. V tej zvezi se širijo govorice, da se IBM sploh ne misli pojaviti na trgu s sistemi, ki uporabljajo procesor 80386 s taktno frekvenco 33 MHz.

A. P. Železnikar

Problemi z OS/400 se nadaljujejo

Napake v IBMovem računalniku srednje velikosti AS/400 spravljajo v obup uporabnike, saj se s t.i. IBMovim "cepljenjem" pojavljajo še dodatni problemi. S paketom Program Temporary Fix (PTF) nudi IBM uporabnikom možnost, da popravijo napake operacijskega sistema AS/400. Vendar je instalacija PTFa dolgotrajna, napake pa so tudi na traku, ki nosi ta paket; tako so učinki še poraznejši kot prej. Pri regularnem odpravljanju napak je potrebno sistem tudi za 24 ur ustaviti in proces s PTF mora biti nadzorovan z živim operaterjem. Pribiližno vsakih šest tednov izda IBM tudi nov PTF trak. Očitno PTFi še niso zadovoljivo dozoreli produkti. Število potrebnih instalacijskih korakov naj bi se zmanjšalo iz začetnih 28 na dva koraka. V nekaterih primerih je npr. PTF tudi porušil celotne dele operacijskega sistema. Problematična je zlasti verzija PTF 1.2.

A. P. Železnikar

Japonska čaka svojo priložnost na trgu programske opreme

Zapadni tržni opazovalci opozarjajo na ofenzivo, ki prihaja z Daljnega vzhoda. Japonska softwarska industrija se pripravlja za napad na zapadna tržišča. V svetovnem merilu je položaj ZDA na področju programske opreme drugim zaenkrat nedosegljiv. Ameriška podjetja izdobavijo 70% vseh softwarskih produktov na svetu in so s 50% najpomembnejša uporabniška država. Svetovno tržišče, ki je bilo za leto 1988 ocenjeno na $50bn, naj bi po mnenju ekspertov do leta 2000 eksplodiralo na več kot 1000 milijard dolarjev ($1000bn).

Japonci so doslej že ugotovili svoje slabosti na področju programske opreme in ob primerni organizaciji in razpoložljivem kapitalu bi se položaj lahko hitro spremenil. Fujitsu je že leta 1983 ustanovil svoje podjetje v Silicijski dolini, kasneje pa še Sony v Palo Alto, Hitachi v San Bruno in Ricoh v Santa Clari. Japonska aktivnost se običajno dolgo časa skriva in je ni mogoče ocenjevati. Podjetja na Daljnem vzhodu nenehno povečujejo napore za obvladovanje proizvodnje programske opreme. Hitachi usmerja že 30% raziskovalnih in razvojnih aplikacij v software. Toshiba je pred kratkim odprla tovarno softwara s 3000 programirnimi specialisti za razvoj komercialnih in industrijskih programov. Tudi NEC namenja $400m letno v razvoj programske opreme.

Ameriški eksperti predvidevajo, da bodo Japonci in drugi Azijci stopili na trg softwara s strategijo treh poti: najprej prek hardwara za odpiranje trga, potem prek sistemskega softwara tja do uporabniške programske opreme. Pri tem jim lahko pomaga tudi Unix. Samo Sanyo ima že 100 tehniških in grafičnih paketov za svoje delovne postaje, ki jih ponuja v okolju Unixa. Pri vsem tem pa bo treba premagati še marsikatero oviro. Ovire pa niso le vprašanja jezika in razlike v mentaliteti, temveč tudi izobraževalni problemi. Izračunali so, da bi Daljni vzhod potreboval leta 1990 že 1,6 milijona sodelavcev, na razpolago pa jih bo le pičel milijon.

A. P. Železnikar

Deja vu videotexa

Beli dokument Evropske skupnosti vsebuje med drugim tudi načrt, kako naj bi se računalništvo in informatika na velikem skupnem trgu uveljavljala z uporabo videotexa pri vsakdanjem nakupovanju. Do nedavnega je ameriški potrošnik odklanjal videotex, ki je bil v ZDA že v letu 1980 uveden kot informacijska storitev. Videotex je npr. propadel tudi v Španiji. V zadnjem času pa poskušata trgovsko podjetje Sears in IBM spremeniti to odklonilno držo Američanov z novimi podatkovnimi in vizualnimi možnostmi pri nakupovanju.

Sears je največja ameriška trgovska hiša za prodajo na drobno. IBM je največje računalniško podjetje na svetu. In ti dve podjetji sta zdaj povezali svoje marketinške in tehnološke mišice s ciljem, da zgradita nov sistem elektronske pošte za informiranje in nakupovanje. Ime tega projekta je Prodigy (slovensko: čudo). Prodigy združuje besedilo in grafiko v obliki živega prikaza in prinaša novice, vremensko stanje in napoved, nakupovanje, razvedrilo, izobraževanje, potovanje, finančne storitve in časopisne izvlečke na dom ali v službo prek PCja. Trenutno je Prodigy na razpolago le na regionalnih trgih, že drugo leto pa bo dostopen na celotnem ozemlju ZDA.

Seveda je zanimiva tudi cena. Za programsko opremo, identifikator in prosti dostop za prve tri mesece bo potrebno odšteti $49,95. Enak paket, toda z modulom za 1200 baudov, ki se ga vstavi v PC, ima ceno $149,95. V tej ceni je zajetih šest družinskih članov, od katerih ima vsak svoj identifikator. Konfiguracija PCja pa je tale: IBMovsko združljivi PC mora imeti vsaj 512k RAMa, grafični adapter, diskovni pogon in že omenjeni modem za 1200 baud. Seveda pa bo mogoče uporabljati tudi Applove in Macintoshove računalnike. Ali bo Prodigy uspel tam, kjer so drugi že propadli?

A. P. Železnikar

Virusna histerija

Kako se je mogoče izogniti virusni histeriji, ki je zajela tako uporabnike PCjev kot mainframov? Virusni programi se pojavljajo v različnih izvedbah in tudi na različnih koncih sveta, od ZDA do Sovjetske zveze. Pred to nadležno in predvsem škodljivo infekcijo se je mogoče obvarovati s posebno zaščitno strategijo, s posebno

kontrolo sistemskih in uporabniških programov preden jih spustimo v obratovanje na naših računalnikih.

Jet Propulsion Laboratory (JPL) v Passadeni (Kalifornija) se je v preteklem letu okužil s štirimi vrstami virusov, ki so napadle tako trdne diske kot tudi povzročile prekinitve dela. Pri virusni zaščiti priporoča JPL dovolj pogosto shranjevanje (backup) in testiranje naložene programske opreme s programi za detekcijo virusov pred njihovo uporabo. V tem primeru so lahko softwarski virusi manj učinkoviti v svoji razdiralni funkciji.

V decembru 1988 se je v IBMovi korporativni mreži pojavil virus, ki se je javljal na zaslonih z božičnimi čestitkami. Programski virusi se pri nekaterih uporabnikih razumevajo podobno kot terorizem in AIDS. Ker so računalniki povezani v mreže, nastaja občutek, da lahko slučajen kontakt ob nepravem času povzroči katastrofo. Najbolj znani primer, ki je bil podoben katastrofi, se je primeril v vladni mreži v novembru 1988, povzročil pa ga je študent R. Morris s Cornellske univerze. Ta virus sicer ni uničeval podatkov, se je pa tako hitro razmnoževal, da so bile mreže prek celotnega kontinenta že v nekaj urah preobremenjene. Če bi bil ta virus konstruiran še za rušenje podatkov, bi lahko povzročil maščevalno opustošenje v računalniških sistemih širom po deželi.

Zanimivo pa je, da Morris ni bil obtožen za kriminalno dejanje in obstaja dvom o obstoju možnosti, da bi ga po obstoječi ameriški zakonodaji sploh bilo mogoče obtožiti. Del teh težav izvira iz dejstva, da vsebujejo tudi legalni programski paketi "viruse". To so lahko nenamerni (tudi neidentificirani) virusi, ki se seveda razlikujejo od zlonamernih. T.i. 13. virus programa Friday (ki je uničil podatke na osebnih računalnikih v Izraelu, Londonu in v JPL) lahko postane bolj splošen. Nekateri so prepričani, da je ta virus le del ledene gore. Prihaja doba t.i. usmerjenih virusov in ta virusni pojav je praktično neomejen. Usmerjeni virusi se oblikujejo s posebnimi nameni in se sčasoma tudi spreminjajo. Specialisti za virusno zaščito poročajo o virusih, ki lahko porušijo korporativno mrežo v določenem časovnem intervalu in povzročijo izgubo zaradi padca produktivnosti korporacije. Pred ameriškimi sodišči pa se že pojavljajo primeri, s katerimi se izterjuje odškodnina, ki jo je kdo utrpel zaradi namernega virusa. To seveda pomeni, da programska oprema vobče ne bi smela biti zaščitena z virusnimi programi, ali pa vsaj ne s takimi, ki lahko poškodujejo druge programe in podatke.

Piratska programska oprema je drugi nepredvidljivi izvor virusov, ki ga praktično ni mogoče kontrolirati. Eden od najbolj razširjenih virusov je nastal v računalniški trgovini v Lahore (Pakistan), kjer sta dva brata, Pakistanca vgradila virus v piratske kopije Lotusa 1-2-3, WordStara in v druge popularne aplikacije, z namenom, da kaznujeta tiste, ki kupujejo piratski software. V tej trgovini so kupovali programsko opremo turisti in poslovneži, ki so obiskali Lahore. Nekateri virusi so lahko tudi takšni, da se oprimejo konkurenčnega softwara, ko so bili vnešeni v sistem in s tem okrnijo ugled proizvajalca izvirne programske opreme. Eden prvih komercialnih paketov z virusom je bil Freehand podjetja Aldus Corp. za Macintosha. Ta virus sporoči prek zaslona vsem Macintoshovim uporabnikom mir na svetu, potem pa izgine. Drugi virus napade CD-ROM in uniči podatke. Freehandov virus je dobil ime mirovni virus in je šolski primer, kako se lahko virus hitro razširja. Ta virus je konstruiral D. Davidson v Tucsonu (Arizona) in ga poslal R. Brandowu, ki ga je spravil na diskete z igrami v Macintoshovi skupini uporabnikov. Kmalu je bilo s tem virusom okuženih na tisoče disket.

Seveda je vprašanje, kakšni so virusni simptomi. Biološki virus povzroča npr. nahod in vročino. Pri računalniškem virusu se zgodi nekaj podobnega. Ti simptomi so lahko upočasnjeno delovanje računalnika, saj veliko število virusov znižuje zmogljivost, ko se dovolj razmnožijo. Drugi opozorilni znaki pa so: dostopanje na disk brez zahteve in nepredvideno zmanjšanje razpoložljivega pomnilnika. Uporabniki sistemov z DOS naj bi kot prvo zaščitno operacijo izvajali program CHKDSK. Ta storitev pove, koliko prostora na trdnem disku je v uporabi, koliko je skritih zbirk in koliko je zgubljenih skupin (clusters). Če se kateri od teh podatkov naenkrat spremeni, je velika verjetnost, da je na delu virus. Programi za detekcijo virusov, ki so na trgu, so primerni za opazovanje virusov. Ti paketi zagotavljajo, da bodo z njimi virusi odkriti in tudi odstranjeni. Seveda pa to velja le za najbolj znane oziroma razširjene viruse.

Oglejmo si nekaj komercialnih paketov za detekcijo in odstranjevanje virusov. MultiPlus je v ROMu rezidentna storitev podjetja SunFlex Software (Atlanta), ki je splošna programska oprema za opravljanje virusne detekcije. Mace 5 podjetja Mace Utilities vsebuje kodno zaščito proti virusom. Paket Murray podjetja Ernst & Whinney nudi širšo zaščito, kot je zaščita z geslom in zaklepanje zbirk vključno z virusno detekcijo. Dobijo se tudi zaščitni paketi, ki se prodajajo skupaj z vsakim trdnim diskom (Arche Technologies). Program naredi virusni preizkus ob vsaki vključitvi računalnika in odstrani najdene viruse. Immune System pa je že računalnik s procesorjem 80286, ki je virusno varen. Ima specialno zaščiteno jedro operacijskega sistema, ki onemogoča spremembe v DOS in BIOS, uporablja geslo in program za zbirčno omejevanje (American Computer Security Industries). Podobna naprava za obstoječe računalnike je plošča Immunetec PC (Zeus Corp.), ki preizkuša sistemske zbirke DOSa in nalagalni sektor trdnega diska na kakršenkoli virus. Immunetec PC omogoča sistemsko administracijo z namenom, da preprečuje nalaganje z disket in da postavlja gesla in avtorizacijske ravnine. Drugi ploščni produkt je Trispan (Micronyx), ki zagotavlja virusno zaščito posameznim uporabnikom v mreži. Trispan preizkuša sistemske zbirke na spremembe zaradi virusa in nudi zaščito, kot je krmiljeni mrežni dostop, gesla, šifriranje in revizija uporabniških poti.

Priporočila, ki se v povezavi z virusno programsko opremo pojavljajo, so kratko tale:

1. Programsko opremo iz t.i. javne domene (tudi iz divjega trga) preizkusi s programi za detekcijo virusov pred uporabo na lastnem sistemu.

2. Shrani izvirne aplikacijske diskete tako, da se ne morejo okužiti.

3. Opravljaj pogosto rezervno shranjevanje (backup).

4. Čimbolj omeji izmenjavo disket, ki vsebujejo izvršljivi kod (npr. zbirke tipa .exe in

5. Postavi pisalno zaščito (write-protect

tab) na vse diskete.

6. Nikoli ne nalagaj sistem s trdnim diskom iz diskete, ki ni originalna ali iz diska, ki ni pisalno zaščiten.

7. Virusna varnost (varnost pred virusi) naj bo le razširitev splošne varnosti; ne obravnavaj je kot posebno varnost.

8. Poučenost in zavest o virusnem problemu sta najboljša pot, da se s tem problemom soočiš.

A. P. Železnikar

Programska in aparaturna oprema za virusno zaščito na IBMovskih PCjih

Certus ($189) je paket za DOS, s katerim se oblikuje mojstrska zbirka vseh zbirk in ta kopija se primerja kasneje z zbirkami ne disku. Program zapisuje spremembe, ki indicirajo viruse in sporoča rezultate. Iz tega je mogoče sklepati, kdaj in kje se je virus pojavil. Certus tudi blokira tiste pisalne dostope, ki naj bi se opravili mimo DOSa. Približno 8k programa je stalno v RAM pomnilniku. Proizvajalec: Foundation Ware, 13110 Shaker Sq., Cleveland, OH 44120.

C-4 ($40) je paket za DOS in je stalno v RAMu. Opazuje znamenja prisotnosti virusov in takoj zamrzne program, ko zazna virus. Pri tem identificira območje delovanja virusa. Po tej transakciji je mogoče delo nadaljevati ali pa ustaviti sistem in ukrepati v smislu odstranitve virusa. Drugi program podjetja InterPath, ki se imenuja Tracer ($50), zapisuje sistemsko informacijo, in sicer status zbirk CONFIG.SYS, kah. Obvešča o spremembah, takoj ko jih najde. Proizvajalec: InterPath, 4423 Cheeney St., Santa Clara, CA 95054.

Disk Watcher ($100) preizkuša na viruse v okviru DOS na dva načina. Najprej naredi preskus na prisotnost virusa pri vsakem vklopu sistema. Nato preide v poseben pomnilniško rezidenten način (zesede približno 50k RAMa) in kontinuirano opazuje sistem glede na viruse. Disk Watcher razpolaga tudi s storitvenimi funkcijami, ki npr. preprečujejo nepričakovano formatiranje in brisanje zbirk. Proizvajalec: RG Software Systems, Inc., 2300 Computer Ave., Suite I-51, Willow Grove, PA 19090.

MultiPlus ($99) je rezidentna storitev (do 100k) za IBMovski PC, ki vsebuje program za detekcijo in odstranjevanje virusov. Storitev vključuje besedni procesor, zbirčni upravljalnik, avtomatični izbiralnik, razmeščevalnik in kalkulator. Virusni del programa identificira programe, ki dostopajo v .EXE in .COM zbirke ali na trdni disk. Pri zaznavi virusa se sproži sporočilo in navodilo za nadljnjo akcijo. Proizvajalec: SunFlex Software, 1447 Peachtree St., Suite 503, Atlanta, GA 30309.

Virus-Pro ($50) je zaščita pred virusi v več stopnjah, in sicer z zapisovanjem statusa DOS zbirk (tipa .EXE in drugih) na originalnih disketah in na trdnem disku. Program primerja stanje originalnih in kasnejših zbirk in če najde razliko, generira sporočilo o spremembah. Proizvajalec: International Security Technology, Inc., Suite 1710, 515 Madison Ave., New York, NY 10022.

Immunetec PC ($295) je plošča za IBMovski PC ki preizkuša DOS sistemske zbirke in nalagalni sektor trdnega diska na prisotnost virusov. Ker je kompatibilna z Novell, 3Com in IBM token ring vezji, omogoča ta plošča sistemskemu administratorju, da ta zaščiti sistem pred nalaganjem z diskete. Administrator lahko postavlja tudi gesla in avtorizirane ravnine dostopa. Zeus namerava uvesti verzijo plošče, ki bi bila kompatibilna tudi z IBMovim PS/2. Proizvajalec: Zeus Corp., 538 Palisades Dr., Akron, OH 44303.

Trispan ($895) je IBMovska plošča za preizkušanje sistemskih zbirk, ki se spremenijo zaradi virusov. Obstajajo tudi drugi varnostni preizkusi vključno z možnostjo zaščite z gesli, šifriranjem in oblikovanjem revizijskih sledi na sistemu ali v mreži. Proizvajalec: Micronyx, Inc., 1901 N. Central Expressway, Richardson, TX 75080.

A. P. Železnikar

Natančnejša virusna detekcija

Kakšna je metoda za natančnejše preskušanje zbirke command.com? Pri tem se lahko uporabi storitev za primerjavo DOS zbirk z imenom fc.exe. Najprej kopirate vašo originalno zbirko command.com na vaš trdni disk, in sicer pod modificiranim imenom, kot je npr. command.tst. Potem dodate posebno vrstico na koncu vaše zbirke (z uporabo EDLIN ali drugega besedega procesorja, ki generira ASCII zbirke), in sicer

```
FC/B COMMAND.COM COMMAND.TST
```

T.i. /B stikalo, ki povzroči binarno primerjavo, sicer ni nujno, ker ima ena od primerjanih zbirk že sufiks .com, tako da bi se binarna primerjava tako ali tako izvršila.

Vsakokrat ko pri vklopu naložite vaš sistem, bo ta ukaz primerjal ti dve zbirki zlog za zlogom. Če sta enaki, se bo pojavilo sporočilo

```
FC: No difference encountered
```

Če pa so razlike, bodo te prikazane. Druga zaščitna mera proti virusom je izvajanje komunikacijskih programov iz t.i. RAM diska. Vse, kar se naloži prek modema, se najprej shrani v RAM disk in od tu se kopira na disketo. Nikoli naj ne bi kopirali naloženo zbirko na trdni disk, dokler niste prepričani, da je zbirka brezhibna. Te, dokaj enostavne zaščitne mere niso absolutne, nudijo pa lahko kar občutno zaščito pred virusi.

A. P. Železnikar

Avtorsko stvarno kazalo časopisa Informatica, letnik 13 (1989)

Č L A N K I

Authors' Subject Index of the Journal Informatica, Volume 13 (1989)

F O R U M I N F O R M A T I O N I S

N O V I C E I N Z A N I M I V O S T I

Z ekspertnim sistemom proti propadanju podjetij (Prenos znanja v računalnike) (A.P. Železnikar) Informatica 13 (1989), No. 3, 68-69.

Zakaj se inženirji zavzemamo za tehnološko tržno združevanje (R. Piskar) Informatica 13 (1989), No. 4, 82-83.

S T R A T E G I J A R A Č U N A L N I Š T V A

Ali IBM lahko postane največje telefonsko podjetje na svetu (A.P. Železnikar) Informatica 13 (1989), No. 4, 72-73.

Digital Equipment v letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 78.

Finska Nokia na pohodu (A.P. Železnikar) Informatica 13 (1989), No. 4, 81.

Francoski Groupe Bull v letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 80.

Fujitsu v preteklem letu (A.P. Železnikar) Informatica 13 (1989), No. 4, 78-79.

IBM v preteklem letu (A.P. Železnikar) Informatica 13 (1989), No. 4, 77-78.

Izgubarji in nazadovalci v računalniški industriji v letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 76.

Kovanje denarja z delovnimi postajami ali Sun Microsystems v letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 81.

Največji prihodek na zaposlenega v računalniški industriji leta 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 76-77.

Poslovni podatki za računalniška podjetja na svetu v koledarskem letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 75-76.

Prestrukturiranje podjetja Olivetti glede na novo realnost (A.P. Železnikar) Informatica 13 (1989), No. 4, 79-80.

Računalniška podjetja z največjo donosnostjo v letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 77.

Računalniška podjetja z največjo rastjo prihodka v letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 77.

Siemens v letu 1988 (A.P. Železnikar) Informatica 13 (1989), No. 4, 79.

Trije trgi oblikujejo računalniško industrijo (A. P. Železnikar) Informatica 13 (1989), No. 4, 73-75.

T O R K O V I I N P E T K O V I P O G O V O R I

Kako so nastali torkovi pogovori. Informatica 13 (1989), No. 3, 82.

Petkovi pogovori. Informatica 13 (1989), No. 4, 95.

Terčelj, A., Grafični standardi v luči uporabniške povezave. Informatica 13 (1989), No. 3, 82.

Železnikar, A. P., Razvojni credo Slovenije in Iskre Delte. Informatica 13 (1989), No. 3, 79-82.

I N F O R M A C I J S K A K O Z E R I J A

Železnikar, A.P., Na rob zgodbi o ražunalniškem razumevanju. Informatica 13 (1989), No. 3, 83-84.

S O M E R E C E N T L Y P U B L I S H E D P A P E R S I N F O R E I G N P R O F E S S I O N A L P E R I O D I C A L S

Informatica 13 (1989), No. 2, 82.

Informatica 13 (1989), No. 3, 84.

Informatica 13 (1989), No. 4, 95.

P O P R A V K I

Popravek naslova avtorja B. Jereba. Informatica 13 (1989), No. 4, 95.

- VME – modul s procesorjem 80386 (D. Novak, Mikra)
- računalniške mreže v Sloveniji, SFRJ in Evropi (A. Dolničar)
- komercialne možnosti uporabe nevralnih mrež in proizvodnje nevronskih računalnikov (S. Jeram)
- grafični standardi (dopolnitev) (D. Pungerčar)

Drugi pogovor (16. junija 1989) je obravnaval tele teme:

- informacijska tehnologija v Španiji (A. P. Železnikar)
- računalniško podprto projektiranje tiskanih vezij na miniračunalnikih (S. Jemec)
- dodatna orodja za projektiranje tiskanih vezij (M. Kovačević, Mikra)

Pogovora je moderiral prof. dr. Anton P. Železnikar, udeležili pa so se ju uslužbenci delovnih enot razvoja, raziskav, prodaje, vodstva in organizacije in zunanji sodelavci.

Popravek naslova avtorja B. Jereba

V naslovu članka "Merjenje paralelnosti algoritmov", Informatica 13 (3) 46-49 je bila ob avtorju B. Jerebu objavljena DO Institut J. Stefan. Delo, ki ga obravnava članek, je nastalo pri usposabljanju raziskovalca B. Jereba na IJS, dočim je naslov njegove matične DO Gorenje Raziskave in razvoj. Za nastalo nelogičnost se uredništvo opravičuje avtorju in njegovi DO.

Da tudi v prihodnje ne bi prihajalo do zagate, ki bi se lahko pojavila ob tako dolgem pojasnilu v samem naslovu, se avtorji naprošajo, da daljša pojasnila postavijo v opombo na prvi strani prispevka.

Urednik

Attention to the Readers of Informatica

Readers of Informatica are kindly requested to send information on their papers recently published in foreign professional (scientific, also philosophical) journals (periodicals). Such information will be regularly published within this column.

Petkovi pogovori

Petkovi pogovori so nadaljevanje torkovih pogovorov in so strokovni informativni forum Poslovne enote za razvoj, raziskave in innovacije Iskre Delte, na katerem teče pogovor o najnovejših dosežkih računalništva in informatike po svetu in v podjetju, in sicer z vidika tehnologije, uporabe in znanosti. Opravljena sta bila le dva petkova pogovora.

Prvi pogovor (12. maja 1989) je obsegal naslednje teme:

- DECova in IBMova strategija mrež standarda OSI (A. P. Železnikar)

Some Recently Published Papers in Foreign Professional Periodicals

Z. Brezočnik and B. Horvat, Automatic Formal Verification of Digital Systems Using Prolog. ACM/SIGCHI *19 (1988), No. 4, 13 – 14 (ACM Press).*

Informatica

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the Slovene Society Informatika, Iskra Delta Computers, Stegne 15C, 61000 Ljubljana, Yugoslavia.*

Volume 13 Number 4 October 1989 YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

CONTENTS