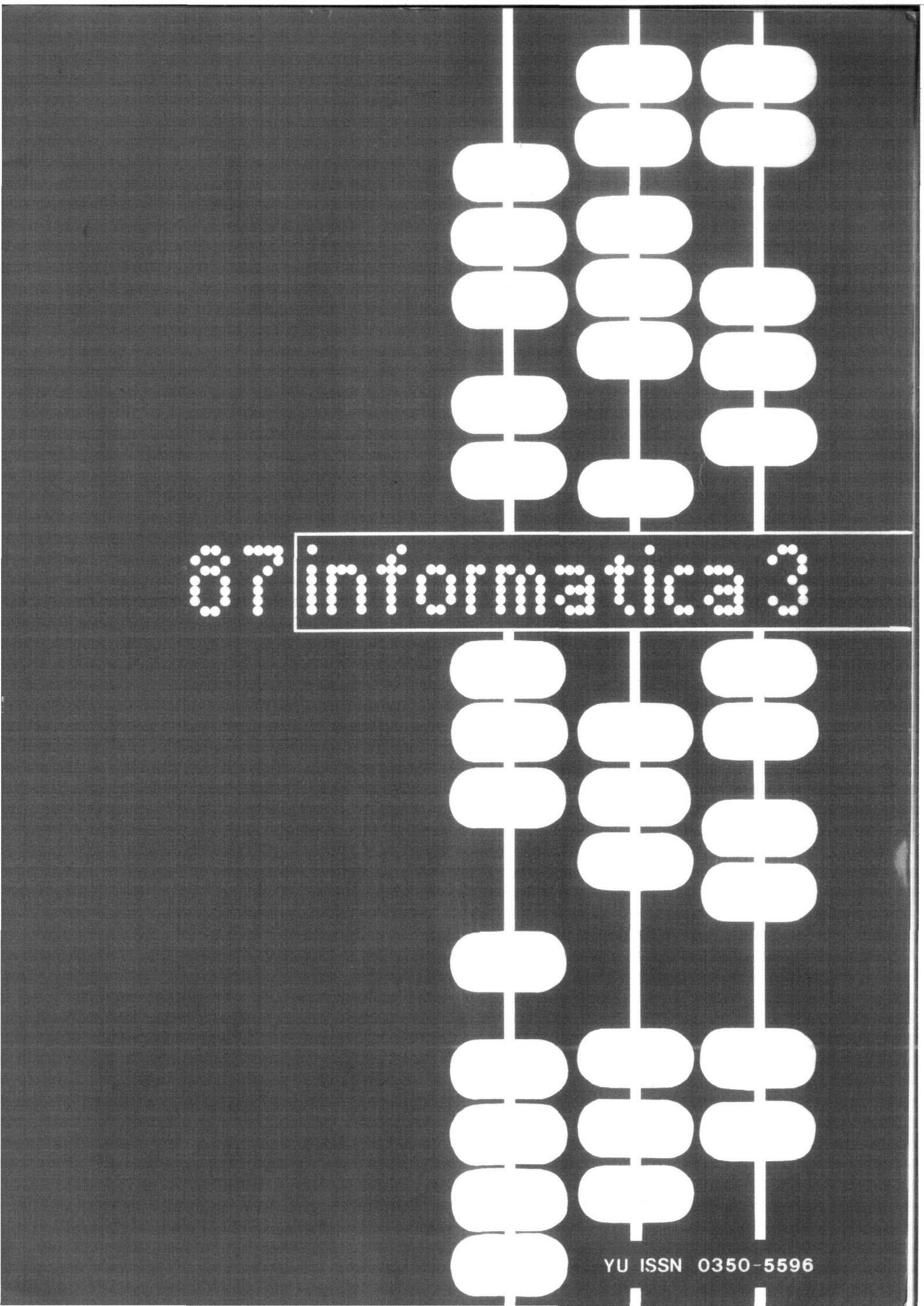


YU ISSN 0350-5596

Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta
Iskra Delta

proizvodnja računalniških sistemov in inženiring, p.o.
61000 Ljubljana, Parmova 41
telefon: (061) 312-988
telex: 31366 YU DELTA

informatica

JOURNAL OF COMPUTING AND INFORMATICS

Published by Informatika, Slovene Society for Informatics, Parmova 41, 61000 Ljubljana, Yugoslavia

YU ISSN 0350-5596

VOLUME 11, 1987 – No. 3

Headquarters:

Informatica, Parmova 41, 61000 Ljubljana, Yugoslavia. Phone: 61 31 29 88. Telex: 31366 yu delta

Annual Subscription Rate: US$ 30 for companies, and US$ 15 for individuals

Printed by: Tiskarna Kresija, Ljubljana

CONTENTS

informatica

ČASOPIS ZA TEHNOLOGIJO RAČUNALNIŠTVA
IN PROBLEME INFORMATIKE
ČASOPIS ZA RAČUNARSKU TEHNOLOGIJU I
PROBLEME INFORMATIKE
SPISANIE ZA TEHNOLOGIJA NA SMETANJETO
I PROBLEMI OD OBLASTA NA INFORMATIKATA

Casopis izdaja Slovensko društvo Informatika, 61000 Ljubljana, Parmova 41, Jugoslavija

Uredništvo in uprava:

Informatika, Parmova 41, 61000 Ljubljana, telefon (061) 312 988; teleks 31366 YU Delta.

Letna naročnina za delovne organizacije znaša 11990 din, za zasebne naročnike 2990 din, za študente 990 din; posamezna številka 4000 din.

Številka žiro računa: 50101-678-51841

Pri financiranju časopisa sodeluje Raziskovalna skupnost Slovenije

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

Tisk: Tiskarna Kresija, Ljubljana

YU ISSN 0350-5596

LETNIK 11, 1987 – ŠT. 3

V S E B I N A

CATEGORICAL APPROACH TO THE RELATIONAL MODEL OF DATA

INFORMATICA 3/87

UDK 519.713

Mara Alagić & Suad Alagić
University of Sarajevo

Abstract. The categorical approach to the basic concepts of the relational model of data is presented.It is shown that the full family of functional dependencies, when viewed appropriately, is in fact a category. This category (of intentions) is the domain of an appropriately defined functor which captures the notion of an extension.Extensions are objects of yet another category, whose arrows are database updates - natural transformations of category theory. Furthermore, the natural join of two relations is proved to be a categorical construction known as the pullback (of a pair of suitably chosen arrows). Categorical generalizations of some familiar results in the theory of the relational databases are proved.

Sažetak. Kategorijski pristup relacionom modelu podataka predstavlja jedan formalni pogled na teoriju podataka. Pojmovi su definisani strelicama i komutativnim dijagramima. Puna familija funkcionalnih zavisnosti,posmatrana na odgovarajući način je kategorija (intencija) i domen je pogodno definisanog funktora koji odgovara pojmu ekstenzije.Ekstenzije su i objekt druge kategorije, čije su strelice ažuriranja baze za koje je pokazano da su prirodne transformacije teorije kategorija.Prirodno spajanje relacija poznata je kategorijska konstrukcija spajanja (pullback) nad parom pogodno izabranih strelica. Dokazana su kategorijska uopštenja nekih poznatih rezultata u relacionom modelu podataka.

Introduction

One of the major advantages of the relational model of data over the models which preceeded it is the higher level of abstraction in viewing the objects in the model and the operations upon them. Rather than providing primitives for operating upon particular data elements, the relational model of data contains operations upon sets of n-tuples and in such a way it provides a higher level, nonprocedural (or at least much less procedural) users' interface. But the way the theoretical foundation of the model is commonly presented is still based on defining the basic concepts in terms of elements.

In this paper we show that the relational model of data may be founded in a rather different,more abstract fashion, in which all the fundamental concepts may be defined in terms of arrows and commutative diagrams with virtually no need for referring to particular data elements. Our approach is based on some basic and fairly simple concepts of category theory introduced in section 1. Further, we present our approach to the functional dependencies. We show that the functional dependencies are not functions, but rather arrows of a category which differs considerably from the category of sets and functions. This category is constructed on the basis of a given set of attributes and functional dependencies among them. Its properties and the relationship to the full family of functional dependencies are analysed in the paper.

Research presented in the paper was supported by Republička zajednica za naučni rad SR BiH.

The fact that the functional dependencies are, in the usual approach, treated as time-varying functions, just as the relations are time-varying sets, is characterized by the categorical notion of natural transformation. The condition that the changes of relations in time must respect given functonal dependencies leads to the characterization of database updates as natural transformations of category theory. While the category of intentions determines the abstract structure of all the relations which may be defined over a given set of attributes and a set of functional dependencies among them, the actual sets which correspond to these abstract relations are specified by arrows whose domain is the category of intentions and codomain is a suitable category for which the category of sets and functions is an obvious choice.

1. The Category of Intentions

A graph consists of a set O of objects, a set A of arrows and two functions: dom: A $\longrightarrow$ O, cod: A $\longrightarrow$ O where "dom" assigns to each arrow f an object X = dom(f) called the domain of f, and "cod" assigns to each arrow f an object Y = cod(f), called the codomain of f. These operations on f are best indicated by displaying f as an actual arrow :

$$f: X \longrightarrow Y$$

and the set of all such arrows f with dom(f) = X and cod(f) = Y is denoted by hom(X,Y).

A category is a graph with two additional functions: Identity, which assigns to each object X an arrow 1_X,

$$1_X: X \longrightarrow X$$

Composition, which assigns to each pair (g,f) of arrows, with dom(g) = cod(f) an arrow fg, called their composite where gf: dom(f) ⟶ cod(g).

The following axioms are required for a category:
Associativity. For given objects and arrows in the configuration

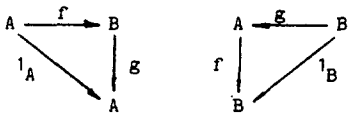
$$X \xrightarrow{f} Y \xrightarrow{g} Z \xrightarrow{h} W$$

one always has the equality h(gf) = (hg)f.
Unit law. For all arrows f: X ⟶ Y and g: Y ⟶ Z composition with the identity arrow 1_Y gives

$$1_Y f = f \text{ and } g 1_Y = g.$$

A familiar example of a category is **Set**. Its objects are sets, its arrows are all functions between them.

In order to illustrate the categorical approach we present the categorical generalization of the notion of isomorphism. Two objects A and B are **isomorphic** in a category C if there exists a pair of arrows f: A ⟶ B and g: B ⟶ A in C such that the following diagrams are commutative:



An isomorphism between objects A and B is denoted by the following: A ≅ B.

An arrow m: A ⟶ B is **monic** in C when for any parallel pair of arrows f_1:D ⟶ A and f_2: D ⟶ A the equality $mf_1 = mf_2$ implies $f_1 = f_2$. In the category **Set** monic arrows are injections, i.e.,the functions which are one - one into.

An arrow h: A ⟶ B is **epi** in C when for any parallel arrows g_1: B ⟶ C and g_2: B ⟶ C the equality $g_1 h = g_2 h$ implies $g_1 = g_2$. In the category **Set** the epi arrows are the surjections, i.e.,the functions onto.

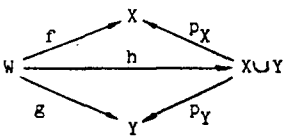
Let A be a finite set of symbols, called attributes. The category of trivial functional dependencies **T** is defined as follows: For each subset X of A there exists an object X of **T**. If X and Y are objects of **T**, then hom(X,Y) consists of only one arrow X ⟶ Y whenever Y is a subset of X, otherwise hom(X,Y) is empty set. If f: X ⟶ Y and g: Y ⟶ Z are arrows of **T** then we obtain X ⟶ Z since Y is a subset of X and Z is subset of Y implies Z is a subset of X so that we have a composite arrow gf, gf: X ⟶ Z. This composition is obviously associative and its identity 1_X: X ⟶ X follows from X ≅ X . So **T** is a category.

The category of intentions I is constructed in three steps starting from the underlying graph of T to which some arrows (nontrivial functional dependencies) are added. Denote the graf obtained that way by G. Before proceeding to the next step we complete the graph G in a crucial way by adding to it an arrow W ⟶ X∪Y whenever G already has arrows W ⟶ X and W ⟶ Y and apply this rule until it does not produce any new arrows of G. In the second step we construct a category whose objects are those of G,whose arrows are finite strings:

$$A_1 \xrightarrow[f_1]{} A_2 \xrightarrow[f_2]{} \cdots \xrightarrow[f_{n-1}]{} A_n \qquad (1)$$

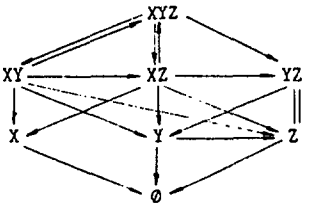
composed of n objects $A_1, A_2, \ldots, A_n$ of G and n-1 arrows f_i: $A_i \longrightarrow A_{i+1}$ of G where each string like (1) is regarded as an arrow $A_1 \longrightarrow A_n$. The composition of these arrows is defined by juxtaposition (concatenation) of strings, identifying the common ends. This composition is associative since the operation of concatenation of strings is. The identity arrows are strings A_n.

The category I constructed in this manner has the following properties:
(i) The category I has finite products,i.e. given objects X and Y there exists an object Z of I with a pair of arrows p_X: Z ⟶ X, p_Y: Z ⟶ Y (called projections) such that for any pair of arrows f: W ⟶ X and g: W ⟶ Y of I there exists a unique arrow h: W ⟶ Z with the property that $p_X h = f$ and $p_Y h = g$. The desired objects Z corresponds to X∪Y and the projections are the unique arrows - trivial functional dependencies. It may be illustrated by the following commutative diagrams:



A generalization to a finite family of objects follows by induction, and instead of X∪Y∪Z∪...∪W it will be denoted by XYZ...V.
(ii) The category I has an initial object,i.e., an object P such that for any other object X there is a unique arrow P ⟶ X.
(iii) The category I has a terminal object,i.e.,an object O such that for any other object X of I there exists a unique arrow X ⟶ O.

Example 1. Let {X,Y,Z} be a given set of symbols. Its power set is {X,Y,Z,XY,XZ,YZ,XYZ} .The underlying graph of T is easy to construct. Suppose that we are given two additional functional dependencies XY ⟶ Z, Z ⟶ Y. Then, the category of intentions contains the category of trivial functional dependencies **T**, XY ⟶ Z, Z ⟶ Y, and derived functional dependencies: XY ⟶ YZ, XZ ⟶ Y, XY ⟶ XYZ, XY ⟶ XZ, Z ⟶ YZ, XZ ⟶ XYZ, XZ ⟶ YZ. Now, the underlying graph of all functional dependencies in I is:



2. The Full Family of Functional Dependencies and the Category of Intentions

Given a set of attributes A, **a full family of functional dependenncies** may be characterized by the following set of axioms (cf. /Codd/) in which X,Y,Z and W are subsets of A, and $X \longrightarrow Y$ denotes a functional dependency:

1. Reflexivity: Y subset of X implies $X \longrightarrow Y$;
2. Augmentation: $X \longrightarrow Y$ and Z subset of W implies $XW \longrightarrow YZ$;
3. Pseudotransitivity: $X \longrightarrow Y$ and $YW \longrightarrow Z$ implies $XW \longrightarrow Z$.

In addition to those axioms functional dependencies may be composed and there may be at most one functional dependency $X \longrightarrow Y$ for any two subsets X and Y of a set A.

It is an usual categorical argument to prove the following:

The category of intentions I satisfies the axioms which characterize the full family of functional dependencies.

3. The Category of Extensions

For categories **B** and **C** a **functor** $K: B \longrightarrow C$ with $dom(K) = B$ and $cod(K) = C$ consists of two suitably related functions: The object function K, which assigns to each object X of **B** an object KX of **C**, and the arrow function (also written K) which assigns to each arrow f, $f: X \longrightarrow Y$ of **B** an arrow $Kf: KX \longrightarrow KY$ of **C** in such a way that:

$$K(1_X) \cong 1_{KX};$$

$$K(gf) \cong KgKf \cong (Kg)(Kf),$$

the latter whenever the composite gf is defined in **B**.

Given two functors $K, L: B \longrightarrow C$, **a natural transformation** $t: K \dot{\longrightarrow} L$ is a function which assigns to each object X of **B** an arrow $tX: KX \longrightarrow LX$ of **C** in such a way that for each arrow $f: X \longrightarrow Y$ in **B**,

$$(Lf)(tX) = (tY)(Kf).$$

We call tX, tY, ... the components of the natural transformation t.

Let **C** and **B** be categories and $K: B \longrightarrow C$ a functor. We say that K **preserves products** if it maps a product $(X \times Y, p_X: X \times Y \longrightarrow X, p_Y: X \times Y \longrightarrow Y)$ in the category **B** into the product in the category **C**, given the object $KX \times KY$ and the corresponding projections Kp_X and Kp_Y, $Kp_X: KX \times KY \longrightarrow KX$ and $Kp_Y: KX \times KY \longrightarrow KY$, i.e., $K(X \times Y) = KX \times KY$.

Given a category of intentions **I**, a **specification of domains** is a functor $D: I \longrightarrow C$, where **C** is a suitably chosen category, which preserves finite products.

The category I of intentions specifies in an abstract fashion all the relations which may be defined over a given set of attributes and which satisfy a given set of integrity constraints (functional dependencies). In order to interpret I in this way one has to specify the domains (i.e., the actual sets) which correspond to particular attributes and subsets of cartesian products of domains (relations) which correspond to objects of I. In addition to this, for a given specification of domains, the actual subsets which correspond to the subsets of the given set of attributes change over time and this also has to be taken into account. These changes are called updates of extensions.

Observe that with the presented definition of specification of domains the functor D specifies not only the domains which correspond to particular attributes, but it also states that a domain of a subset $\{A_1, A_2, \ldots, A_n\}$ of attributes from A is the product of the corresponding domains, namely $D(A_1) \times D(A_2) \times \ldots \times D(A_n)$. Equally important is that the functor D assigns to each functional dependency $f: X \longrightarrow Y$ a function $Df: DX \longrightarrow DY$. Thus D interprets functional dependencies as functions. The actual specification of relations which correspond to particular subsets of A within this framework is performed as follows.

Let I be a category of intentions, and $D: I \longrightarrow$ **Set** a specification of domains. Then an **extension** of I is a natural transformation $e: R \dot{\longrightarrow} D$ where $R: I \longrightarrow$ **Set** is a functor such that each component $eX: RX \longrightarrow DX$, for X object of I is an injective function. Therefore an arrow eX is simply a specification of a subset RX of DX. If X is the set $\{A_1, A_2, \ldots, A_n\}$ then RX is some subset of the product $DA_1 \times DA_2 \times \ldots \times DA_n$. But more than that, e is a natural transformation which means that for each arrow $f: X \longrightarrow Y$ of I the functions Rf and Df are related by the equality $(Df)(eX) = (eY)(Rf)$, which simply means that Rf is a "subfunction" of Df, as it should be. Let X,Y,Z and W be subsets of A such that $W = XYZ$. This means that W is the product of the objects X, Y and Z in I. W is equiped with the projections $p_X: W \longrightarrow X$, $p_Y: W \longrightarrow Y$, $p_Z: W \longrightarrow Z$.

Suppose now that there exists an arrow (functional dependency) $f: X \longrightarrow Y$ in I. We say that the functional dependency f is **embedded into** RW if and only if

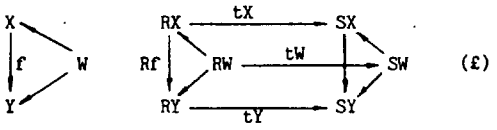
$$(Rf)(Rp_X) = Rp_Y.$$

An essential property of an extension is stated in the following: Let $R: I \longrightarrow$ **Set** be an extension, and X,Y and W objects of I such that X and Y are subsets of W. Then if there exists an arrow $f: X \longrightarrow Y$ in I it is embedded into RW.

More precisely we would like to have $(eW)(Dp_X) = (Rp_X)(eX)$, $(eW)(Dp_Y) = (Rp_Y)(eY)$, $(eW)(Dp_Z) = (Rp_Z)(eZ)$ where $eW: RW \longrightarrow DW$, $eX: RX \longrightarrow DX$, $eY: RY \longrightarrow DY$ and $eZ: RZ \longrightarrow DZ$ are the injections. But e is a natural transformation and for given functional dependency p_X, $p_X: W \longrightarrow X$, the above equalities are valid. Therefore, the images under an extension of trivial functional dependencies are preciselyrestrictions of projections.

4. Natural updates and the Category of Extensions

For a given specification of domains D: I ⟶ Set an extension R ⇢ D is a specification of the actual relations at a given point in time. At some other point in time these relations (i.e. sets) are different due to updates which occured in the meantime. These updates are such that they must respect functional dependencies. More precisely, if R ⇢ D and S ⇢ D are extensions which correspondto these two points of time, then we define an **update of extensions** as a natural transformation R ⇢ S. This definition means that t: R ⇢ S translates the picture of I under R into a picture of I under S. These pictures, of course, lie in Set, as shown in the following diagrams, where X, Y and W are objects of I such that X and Y are subsets of W:



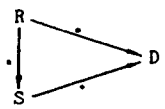
(£)

All the above diagrams (triangular and rectangular) are commutative under the adopted assumption and thus we have the following:

Let R: I ⟶ Set and S: I ⟶ Set be extensions and t: R ⇢ S a natural update. Then if f: X ⟶ Y is em - bedded into RW, it is also embedded into SW, and in such a way that all the diagrams (£) are commutative.

As one might have suspected we can construct a cate - gory whose objects are extensions and whose arrows are natural updates in the following precise sense:

For a given category of intentions I and a specification of domains D: I ⟶ Set, there exists a category E whose objects are extensions R ⇢ D, and whose arrows are natural updates t: R ⇢ S such that the following diagram is commutative:



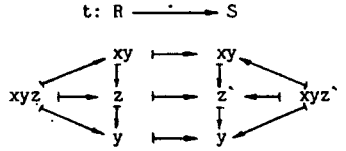
Example 2. Consider again Example 1. Let

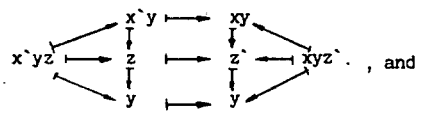
R(XYZ) = { xyz, x`yz, x``y``z`` }, and

S(XYZ) = { xyz`, x`yz` }.

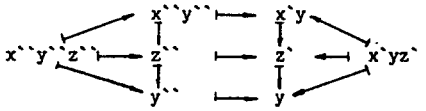
Denote by t a natural update R ⇢ S. There are only two choices for the specification of such an update. One of them is

t: R ⇢ S





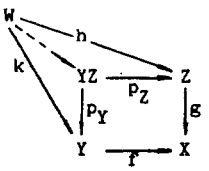
, and



5. Pullbacks and Joins

In this section we show that the join operation of the relational algebra is, in fact, a categorical construction which is called a pullback. The notion of a **pullback** is defined as follows: Given pair of arrows f: Y ⟶ X and g: Z ⟶ X (with a common codomain), a pullback of (f,g) is an object $Y \circ_X Z$ with a pair of arrows p_Y: $Y \circ_X Z \longrightarrow Y$, p_Z: $Y \circ_X Z \longrightarrow Z$ such that $gp_Z = fp_Y$. Furthermore, given any other pair of arrows h: W ⟶ Y and k: W ⟶ Z with gk = fh, there is a unique arrow r: W ⟶ $Y \circ_X Z$ such that $k = p_Z r$ and $h = p_Y r$.

Since we will show that the natural join is the pullback of a pair of suitably chosen arrows, it is certainly of interest the fact that the category of intentions possess pullbacks. Actually, given a pair of arrows f and g, f: Y ⟶ X and g: Z ⟶ X their pullback is the object YZ together with a pair of arrows p_Y: YZ ⟶ Y and p_Z: YZ ⟶ Z which are trivial functional dependencies. That is the consequence of the following commutativities:



We are now in a position to present a generalized de - finition of the join operation of the relational algebra. Given functional dependencies f: Y ⟶ X and g: Z ⟶ X the functional join of f and g, denoted by $R(Y)\circ_X R(Z)$ is defined as the pullback of Rf and Rg.Therefore it consists of an object $R(Y)\circ_X R(Z)$ together with a pair of arrows r_Z: $R(Y)\circ_X R(Z) \longrightarrow R(Z)$, r_Y: $R(Y)\circ_X R(Z) \longrightarrow R(Y)$, such that $(Rg)r_Z = (Rf)r_Y$.

In the usual terminology a natural join of the relations R(XY) and R(XZ) is the set of triples (x,y,z) such that (x,y) belongs to R(XY) and (x,z) belongs to R(XZ). The relations R(XY) and R(XZ) are equipped with (restrictions of) projections p_X: R(XY) ⟶ R(X) and, similarly q_X: R(XZ) ⟶ R(X), where R now denotes an extension.

So, one can prove the following characterization of the natural join: In the category Set the natural join of the relations R(XY) and R(XZ) is preciesly the functional join of the pair of projections p_X: R(XY) ⟶ R(X) and q_X: R(XZ) ⟶ R(X).

Further, it is given a sequence of properties that may be proved using standard categorical tools, having in mind all introduced concepts:

(a) Let X and Y be objects of I such that there exists a functional dependency Y ⟶ X. Then

$R(X)o_X R(Y) \cong R(Y)$

(b) $R(X)o_\emptyset R(Y) \cong R(X) \times R(Y)$

(c) $R(X)o_X R(X) \cong R(X)$ (idempotency)

(d) $R(X)o_A R(Y) \cong R(Y)o_A R(X)$ (commutativity), where two functional dependencies, X ⟶ A and Y ⟶ A, are given.

(e) Given functional dependencies X ⟶ A, Y ⟶ AB and Z ⟶ B, one has associativity in the following form

$$(R(X)o_A R(Y))o_B R(Z) \cong R(X)o_A(R(Y)o_B R(Z)).$$

The relationship between R(XYZ), $R(XY)o_X R(XZ)$ and $R(X) \times R(Y) \times R(Z)$ is the subject of the following (two) propositions:

(1) There exists a canonical way to construct the arrows:

e: R(XYZ) ⟶ $R(X) \times R(Y) \times R(Z)$,

f: R(XYZ) ⟶ $R(XY)o_X R(XZ)$,

g: $R(XY)o_X R(XZ)$ ⟶ $R(X) \times R(Y) \times R(Z)$

such that e = fg and e and g are injective functions.

(2) There is a canonical way to construct injective arrows:

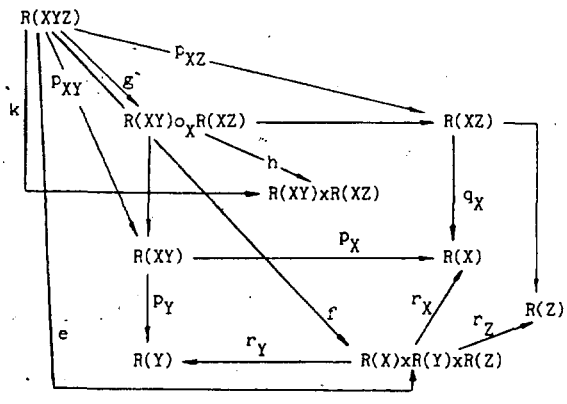
k: R(XYZ) ⟶ $R(XY) \times R(XZ)$,

g`: R(XYZ) ⟶ $R(XY)o_X R(XZ)$,

h: $R(XY)o_X R(XZ)$ ⟶ $R(XY) \times R(XZ)$,

such that hg = k.

In both propositions, the proofs are based on the commutativity of the corresponding diagrams:



6. Equational Properties of Joins

The idempotency, commutativity and associativity of functional joins imply a number of interesting properties out of which, here are selected a few:

(a) Given functional dependencies X ⟶ AB, Y ⟶ AB, Z ⟶ AB, one have

$$(R(X)o_A R(Y))o_B R(Z) \cong R(X)o_B(R(Y)o_A R(Z))$$

(interchange of join attributes).

(b) Interchange of join relations:

Given functional dependencies X ⟶ A, Y ⟶ AB and Z ⟶ AB, one has

$$(R(X)o_A R(Y))o_B R(Z) \cong (R(X)o_A R(Z))o_B R(Y)$$

(c) Given functional dependencies Z ⟶ B and Y ⟶ B,

$$(R(X) \times R(Y))o_B R(Z) \cong (R(X) \times R(Z))o_B R(Y).$$

(d) Given functional dependencies X ⟶ A and Y ⟶ A or Z ⟶ A, the following isomorphism exist:

$$(R(X)o_A R(Y)) \times R(Z) \cong (R(X)o_A R(Z)) \times R(Y),$$

$$(R(X) \times R(Y))o_A R(Z) \cong (R(X) \times R(Z))o_A R(Y),$$

(e) The existence of the functional dependencies Z ⟶ B, X ⟶ B implies

$$(R(X) \times R(Y))o_A R(Z) \cong (R(X)o_A R(Z)) \times R(Y) \text{ and}$$

$$(R(XB) \times R(Y))o_A R(ZB) \cong (R(XB)o_A R(ZB)) \times R(Y).$$

(f) . . .

7. The Universal Relation Model

The universal relation model aims at achieving com - plete access-path independence in relational databases. There is a seqeuence of assumptions that are frequently made implicitly when talking about it. In the categorical approach, those assumptions have the following form:

(1) There is a universal relation scheme whenever a specification of domains is given for the considered category of intentions, i.e. there exists a domain-functor D: I ⟶ **Set**.

(2) Access paths are embedded in attribute names in the following manner: projections in the corresponding category of intentions, from the product-attribute in to the single attribute. For example,

$$NAME \cong N_1N_2\ldots N_k \longrightarrow N_i,$$

where N_i stands to denote different names for different i, i=1,2,...,k (names of employees, customers, suppliers, managers, ...)

(3) For a specific category of intentions there is a unique relationship that the user has in mind. There - fore, it has to be chosen an extension e: R ⟶ D, namely e_X: RX ⟶ DX is the collection which determines the most basic relationship, one the user is interested in, and by that I is "transformed" to the desired relations, and therefore, not only attributes are expected to play unique roles, but combinations of attributes likewise have a unique meaning.

(4) Specifying for every attribute set A a unique access path to compute R(A) does not mean in general that there are not any other access paths for computing R(A).

R(A) may be realized through the join operation, for example, $\underline{R(A)} \cong R(X) o_Z R(Y)$. It is known that the following proposition holds:

$R(XY) \cong R(XZ) o_Z R(YZ)$ if and only if either $Z \longrightarrow X$ or $Z \longrightarrow Y$ exists,

and in such a case $R(A) = \underline{R(A)}$.

Sometimes, this property is called "one flavor assumption".

(5) The assumption that query processing consists of two steps:
i. For the set of attributes mentioned in the query construct e_X: $RX \longrightarrow DX$ ("binding")
ii. Whatever operations must be applied to answer the query are then applied to RX ("evaluation").

8. Concluding Remarks

The result presented in the paper have already been summarized in the abstract and in the introduction. In this concluding section we concentrate on one aspect of the paper which is particularly important. It has not been emphasized throughout the paper since it requires further careful research.

The generality of our approach comes from the fact that the category C does not have to be the category of sets. If it is we obtain the usual approach where relations are simply sets of n-tuples. But it has been pointed out that this is too simple an approach if we want to take into account the possibility that the values of some attributes of an n-tuple may be undefined. That kind of approach (/Codd/, /Vassiliou/) lead to conclusion that it is necessary to impose additional structure upon relations. The approach /Codd/ would, roughly speaking, correspond to taking for C the category of sets with a selected point - the null point (undefined value) and defining the arrows of this category appropriately. If we define its arrows as null - point preserving functions we obtain the category Set_p whose properties have been carefully studied. This category suits nicely for most cases (almost all of the material in the paper applies to it without changes, various types of joins defined in /Codd/ may be captured). But there are the cases in which it turns out not to be quite appropriate (null-points must be preserved, which does not correspond to the usual database updates). So more work is neccessary in order to define appropriately the category C which would cap - ture the approach /Codd/.

Another mathematical approach to the undefined values /Vassiliou/ would correspond roughly to taking complete lattices for the objects of C and continuous (or perhaps just monotone) functions for its arrows. In this approach a considerable amount of structure is added to the sets of n-tuples, but our preliminary investigations show that this category also exibits some undesirable properties (there are problems in constructing pullbacks , i.e., joins). So this aspect requires further investigations too.

We conclude the paper with a remark that one of our sources of motivation for this work was our hope for the development of a general and unified approach which would capture the particular ones mentioned above.

References

/Aho/ Aho A.V.,Beeri,C and Ullman, J.D. The Theory of Joins in Relational Databases, ACM Trans.Database Syst. 4, 3 (Sept.79), 297-314.

/AM/ Alagić, M. Kategorija višeznačnih preslikavanja, Magistarski rad, Beograd (1975),1-84.

/AS/ Alagić, S. Natural State Transformations, Journal of Computer and System Sciences (April 75), Academic Press, 10, 266-307.

/Bernstein/ Bernstein,P.A. Synthesizing Third Normal Form Relations from Functional Dependencies, ACM Trans. Database Syst. 1,4 (Dec.76), 272 -298.

/Codd/ Codd,E.F. Extending the Database Relational Model, Proc. ACM SIGMOD Conf.,Boston 1979, 29-52.

/Fagin/Fagin, R. Normal Forms and Relational Database Operators, Proc. ACM SIGMOD Conf. Boston 1979, 153-163.

/Maier/ Maier, D.,Ullman,J.D., Vardi,M.Y. On the Foundations of the Universal Relation Model, ACM Trans.Database Syst. 9, 2 (June 84) 283-308.

/Mac Lane/ Mac Lane S. Categories for the Working Mathematician, Springer-Verlag, Berlin,Heidelberg, New York, 1971.

/Vassiliou/ Vassiliou, Y. Null Values in Data Base Management. A Denotational Semantics Approach, Proc. ACM SIGMOD Conf., Boston 1979, 162-169.

/AS/ Alagić, S. Relacione baze podataka,Svjetlost, 1985, Sarajevo,1-168.

/AS/ Alagić, S. Relational Database Technology, Springer Verlag,New York, Berlin, Heidelberg, Tokyo,1986, 1-259.

——— : ———

Alagić Mara

Odsjek za matematiku
Prirodnomatematički fakultet
Vojvode Putnika 43
71000 Sarajevo

Alagić Suad

Odsjek za informatiku
Elektrotehnički fakultet
Toplička bb
71113 Sarajevo

UDK 519.72

Anton P. Železnikar
Iskra Delta, Ljubljana

Abstract. This essay deals with principles, consequences, and examples of informational forms and informational processes. First, the principles of informational spontaneity, information, and counter-information are posted and then several principles of information processing are revealed: Informing, embedding, arising, and Counter-Informing of information. The so-called circularity properties of information for these principles are investigated, which explain the circularity, recurrence, parallelness, and serialness (sequentialness) of information. Informational form and informational process are described on the basis of informational principles and informational consequences and as particularly, informationally structured and organized phenomena (informational Enframing). Several informational forms and informational formulae are presented as examples, explicating the possibilities of a soft informational formalization. Examples of intelligence as information, of an informational machine, an informational program, and living information are examined and discussed in this framework of developed principles of information. At the end, a short conclusion concerning formalization of informational principles is given.

> ... The first condition for language to work is that the hearer be open to enter into a conversation with the speaker. No amount of logical or beautiful or stirring language makes sense if it is not listened to. There is no use talking if the backgrounds of speaker and hearer are too different to create open and serious listening - even if the words are read, they will be too distantly reinterpreted.
>
> (Winograd, OUC, p. 257)

0. Introduction

One of the aims of this essay is to open a new conversation, to articulate the power of the human informational mind, and just begin to reveal consequences of some informational principles. This conversation and its power are grounded in the phenomenology of information, in the so-called principles and consequences of information. Information principles and their consequences, both of which are explicated in this essay, have to be understood as an arising beginning, origin, commencement, source, root, supposition, and foundation, as well as the perspective openness of the developing philosophy of information.

In general, natural principles are (logically, ontologically) inconsistent. Does it also hold true for the proposed information principles? In some way, we are always using the faculty of free choice, playing a game with formal or even metaphysical notions and their symbols. Principles are only constructions, for reality has to be constructed, it is not waiting to be discovered (SCC).

The term 'information', its notion, meaning, and understanding are usually accepted as given (ready-to-hand), which does not enter into comprehensional (perceptual and cognitional) details of the informational phenomenology. In some way, the notion of information also becomes a processing and system of information, particularly in the context of computer based man-machine interaction, application, and organization and in the context of cognitive science.

The principled question of information can develop several new questionings which offer new consequences with a sufficient, linguistically formal rigorousness. In this essay, the epistemology of information will not be presented; it was exposed in (OWI) and, in even more detail, in (IDI) . Information possesses a semantically circular and dynamic meaning, and this meaning can be pursued through human history (for example, from the Greek 'eidos', 'morfe', and 'poiesis', to the Latin 'informatio' and today's colloquial meaning of information, informing, informative, informational, information processing, etc.).

In the text which follows, several basic principles of information will be developed in a systematic and linguistically formal way. Through this development, it will become possible to articulate and to explicate the phenomenon of information as the most general experience and as the most regular principle of modern thinking, which is founded on circular comprehension (recurrent observation, investigation, cognition). The forth-coming information era will develop informational thinking and comprehension through information with the intention to structure and to organize a new philosophy and to explore new

informational (intelligent) technological possibilities. On the way to information several new adventures are hidden which have to be investigated in much greater detail.

1. Information and Counter-Information

> ... Every triumphant theory passes through three stages: first it is dismissed as untrue; then it is rejected as contrary to religion; finally, it is accepted as dogma and each scientist claims that he had long appreciated the truth.
>
> (Gould, ESD, quoting embrylogist von Baer)

We experience information as a spontaneously developing phenomenon of thought in our mind. This experience tells us that information is not only developing, spontaneously coming into existence, and arising, but that it is also vanishing, changing, etc. Under these circumstances, information appears to be an extremely free and dynamic process of our mind. This various informational processing is illimitably circular concerning the entire information of a being. In general, we assume that information informs in a spontaneous way, and that on its way to information it is faced with its own observation, investigation, and cognition, which result into a newly arised information, the so-called counter-information. On the way to information, counter-information becomes regular information through its embedding in existing information and yet, regular information qua information is going to inform forward on its subsequent way.

Information qua information informs in several domains. An obviously essential domain of information is the so-called comprehensional domain, by which information observes, investigates, and recognizes itself and other information. From the perspective of this domain, information is in an observant, investigational, and cognitional relation to itself and to other information. According to this comprehensional informational component (domain), information can generate a new, controversial part of itself and of other information, which contradicts the entire existing information and causes the appearance of counter-information. In this manner, the arising counter-information is nothing else than a comprehensional information of information. In general, information is informational and counter-informational simultaneously. Because information is self-comprehensional, it is counter-informational. Potentially, to some extent, information is non-informational or is outside of existing information when it generates counter-information.

On the basis of this short informational investigation, we can state three basic principles of information.

THE PRINCIPLE OF INFORMATIONAL SPONTANEITY. Information is a circularly spontaneous phenomenon of information. Circular informational spontaneity is information by itself.

In this principle, circular spontaneity must be understood as a fundamental informational property, which is in no way limited, bounded, or restricted information and is unforeseeably free in its arising, embedding, counter-Informing, and circulating. Spontaneity also stands outside of and above an autopoietic system which is autopoietically bounded. Usually, spontaneity is understood also as chaos or chance; chaotic dynamical, random, or irregular phenomenon; turbulent flow, unpredictable evolution (of biological species, economic prices), chaotic behavior, irreversible approach (process), irreversible evolution (CCH), etc.

The spontaneous has a condensed meaning of coming into existence, of the volational, free, unexpected. It is not possible to be spontaneous without the coming into existence of something that comes, that takes place. Spontaneous means that it is not known yet what and how the spontaneous will arise, even if one states that something was spontaneous. The 'was' is not spontaneous anymore, it lost its spontaneity as soon as it became a fact. The spontaneous points to the future, for which it is not given in advance what will appear, happen, arise, come into existence, counter-inform, etc. The spontaneous points to 'it-will', however to the yet unknown 'it-will'. In this way, spontaneity, for example, does not concern science (or even ideology), because science can only be the already known. In this respect, science cannot be spontaneous. In contrast to sciences, information's first principle is informationally circular spontaneity which surpasses any principle of truth, falsehood, correctness, identity, objectivism, traditionalism, etc. Spontaneity points to the opening, inkling of the horizon, awareness beyond the horizon of knowing, opening out to new horizons, and standing open.

The spontaneity of information is circular. That what spontaneously arises as counter-information is embedded into existing information and from this information the spontaneous arising proceeds (informs) into a subsequent process. Thus, spontaneity performs circularly. If informational spontaneity designates an informational property (form) of spontaneity, then arising designates an informational process of spontaneity in which information is spontaneously arising (coming into existence).

THE PRINCIPLE OF INFORMATION. Information informs means that information circularly and spontaneously generates information qua counter-information, circularly and spontaneously embeds new information into the existing one, or that information is circularly and spontaneously arising, coming into existence, changing, vanishing, embedding, etc. through information itself. Information possesses the mastery (forming and processing) over itself.

On the basis of this principle a philosophy or theory of information can be developed in a much more detailed form. The observation one can make with this principle is that the circular and spontaneous forming and processing nature of information represents an (mathematically, algorithmically, formally, and even linguistically) unusual problem of formalization standing outside of the so-called rationalistic tradition.

THE PRINCIPLE OF COUNTER-INFORMATION. According to the principle of information, information is coming into existence as counter-information. Counter-information informs as information. Counter-information is coming into existence as a comprehension of information by information, as a regular informational form or informational process. Counter-information is information by itself.

These principles are in no way closed statements because they depend on completely open notions, some of which will be captured in the following principles, consequences, and

the principles of circularity, recurrence, parallelness, and serialness of information (see the following principles). Informing of information is information by itself.

THE PRINCIPLE OF EMBEDDING OF INFORMATION. Embedding of information means the insertion, nesting, or deposition (also interpreting, coding, meaning, understanding, etc.) of some free, arising, counter-informational information into a given comprehensive body or realm of information. Through embedding, information is put into a regular informational context (possible informational relations), into the realm of existing information (possible meaning). On a linguistic level, the semantic relation between new information and existing information is coming into existence. Embedding of information is information by itself.

THE PRINCIPLE OF ARISING OF INFORMATION. Arising of information is the generating, developing, vanishing, and coming of information into existence. Arising of information has the power of new informational appearance, the coming of new information into the present. Informationally, arising of information is not limited in any way; it is circular and spontaneous. Through arising, any information can come into existence. Arising of information is information by itself.

THE PRINCIPLE OF COUNTER-INFORMING OF INFORMATION. Counter-Informing of information means that information, which through its Informing of information, is comprehensively informed. The product of this Informing is counter-information. Counter-Informing is the production of the observant, investigational, and cognitional information in the process of Informing. Counter-Informing of information is information by itself.

THE CONSEQUENCE OF INFORMING. Informing of information is a form and/or a process of circular and spontaneous Informing, embedding, arising, and Counter-Informing of information.

AN EXAMPLE OF INFORMING. The last consequence can be expressed in a formalized way as

```
Informing
   means
Informing, embedding, arising,
   Counter-Informing;
```

This formula is recursive in Informing, embedding, arising, and Counter-Informing. All of these entities are informationally interwoven.

AN EXAMPLE OF INFORMATIONAL EMBEDDING. The principle of informational embedding can be interpreted by the following formula:

```
embedding
   means
      or
   is_equal_to
semantical_connection_of
   counter-information
      and
   existing_information;
```

where each entity is written on a separate line. The metaphorical presentation of this formula explicates the meaning and other possibilities of a formula's expression. In the Polish (prefix) notation this formula is longer:

```
or means
        embedding semantical_connection_of
        and
           counter-information
           existing_information
   is_equal_to
        embedding semantical_connection_of
        and
           counter-information
           existing_information;
```

In this case, the binary (or multiple) *operators* 'or', 'means', 'and', 'is_equal_to' can be understood as parallel operations, 'semantical_connection_of' functions as a unary operator, 'counter-information' and 'existing_information' as parallel processes, and 'embedding' as a parallel, two-part result.

EXAMPLES OF INFORMATIONAL ARISING. Examples of formulae concerning the principle of informational arising are:

(1) The formula of informational deductive inference (implication) is:

```
  Information generates counter-information;
counter-information is_embedded_in information;
-----------------------------------------------
          counter-information becomes
             regular_information;
```

(2)
```
                arise
            has_the_meaning_of
            appear, arise, be,
      bring_to_the_comprehensional_surface,
       come_in_presence, come_into_being,
  come_into_existence, contradict, counteract,
  counter-inform, create, develop, disturb, do,
   emerge, enable, establish, evolve, expose,
     form, generate, grow, imagine, inform,
      innovate, inspire, make, mutate, oppose,
    originate, process_anew, produce, set_up,
         shape, spring_up, unfold, ... ;
```

(3)
```
   information generates information
                is_equal_to
   information is_generated_from information;
```

AN EXAMPLE OF A COUNTER-INFORMING FORMULA. Counter-Informing as information has its origin in Informing of information. It is the comprehensive part of Informing which observes, investigates, and recognizes information and Informing of information. It arises during Informing of information as an informational counter-part. It springs up from Informing or is generated by Informing. Hence, the following *informational formula* is possible:

```
          Counter-Informing
originates_from or is_generated_by
      Informing_of_information
                as
observing, investigating, recognizing
                of
information, Informing of information;
```

Several other formulae of Counter_Informing are possible and can be determined in a more detailed manner.

3. Principles of Informational Circularity

How is information arising in a circular way? What is the nature of informational circularity? How does it concern the realm of information? Some answers to these questions will be given by the subsequent principles, consequences, and examples.

the principles of circularity, recurrence, parallelness, and serialness of information (see the following principles). Informing of information is information by itself.

THE PRINCIPLE OF EMBEDDING OF INFORMATION. Embedding of information means the insertion, nesting, or deposition (also interpreting, coding, meaning, understanding, etc.) of some free, arising, counter-informational information into a given comprehensive body or realm of information. Through embedding, information is put into a regular informational context (possible informational relations), into the realm of existing information (possible meaning). On a linguistic level, the semantic relation between new information and existing information is coming into existence. Embedding of information is information by itself.

THE PRINCIPLE OF ARISING OF INFORMATION. Arising of information is the generating, developing, vanishing, and coming of information into existence. Arising of information has the power of new informational appearance, the coming of new information into the present. Informationally, arising of information is not limited in any way; it is circular and spontaneous. Through arising, any information can come into existence. Arising of information is information by itself.

THE PRINCIPLE OF COUNTER-INFORMING OF INFORMATION. Counter-Informing of information means that information, which through its informing of information, is comprehensively informed. The product of this Informing is counter-information. Counter-Informing is the production of the observant, investigational, and cognitional information in the process of Informing. Counter-Informing of information is information by itself.

THE CONSEQUENCE OF INFORMING. Informing of information is a form and/or a process of circular and spontaneous Informing, embedding, arising, and Counter-Informing of information.

AN EXAMPLE OF INFORMING. The last consequence can be expressed in a formalized way as

Informing
means
Informing, embedding, arising,
Counter-Informing;

This formula is recursive in Informing, embedding, arising, and Counter-Informing. All of these entities are informationally interwoven.

AN EXAMPLE OF INFORMATIONAL EMBEDDING. The principle of informational embedding can be interpreted by the following formula:

embedding
means
or
is_equal_to
semantical_connection_of
counter-information
and
existing_information;

where each entity is written on a separate line. The metaphorical presentation of this formula explicates the meaning and other possibilities of a formula's expression. In the Polish (prefix) notation this formula is longer:

or means
embedding semantical_connection_of
and
counter-information
existing_information
is_equal_to
embedding semantical_connection_of
and
counter-information
existing_information;

In this case, the binary (or multiple) operators 'or', 'means', 'and', 'is_equal_to' can be understood as parallel operations, 'semantical_connection_of' functions as a unary operator, 'counter-information' and 'existing_information' as parallel processes, and 'embedding' as a parallel, two-part result.

EXAMPLES OF INFORMATIONAL ARISING. Examples of formulae concerning the principle of informational arising are:

(1) The formula of informational deductive inference (implication) is:

information generates counter-information;
counter-information is_embedded_in information;
---
counter-information becomes
regular_information;

(2)
arise
has_the_meaning_of
appear, arise, be,
bring_to_the_comprehensional_surface,
come_in_presence, come_into_being,
come_into_existence, contradict, counteract,
counter-inform, create, develop, disturb, do,
emerge, enable, establish, evolve, expose,
form, generate, grow, imagine, inform,
innovate, inspire, make, mutate, oppose,
originate, process_anew, produce, set_up,
shape, spring_up, unfold, ... ;

(3) information generates information
is_equal_to
information is_generated_from information;

AN EXAMPLE OF A COUNTER-INFORMING FORMULA. Counter-Informing as information has its origin in Informing of information. It is the comprehensive part of Informing which observes, investigates, and recognizes information and Informing of information. It arises during Informing of information as an informational counter-part. It springs up from Informing or is generated by Informing. Hence, the following informational formula is possible:

Counter-Informing
originates_from or is_generated_by
Informing_of_information
as
observing, investigating, recognizing
of
information, Informing of information;

Several other formulae of Counter_Informing are possible and can be determined in a more detailed manner.

3. Principles of Informational Circularity

How is information arising in a circular way? What is the nature of informational circularity? How does it concern the realm of information? Some answers to these questions will be given by the subsequent principles, consequences, and examples.

THE PRINCIPLE OF CIRCULARITY OF INFORMATION. Information is spontaneously circular concerning information. Informational circularity concerns the entire realm of information (counter-information, Informing, embedding, arising, Counter-Informing) and is informationally transparent. Different forms and processes of informational circularity are recurrence, parallelness, serialness (sequentialness), and yet not identified circular forms and processes of information. Circularity of information is information by itself.

THE PRINCIPLE OF RECURRENCE OF INFORMATION. Informational recurrence is the information-inherent and information-transparent property of circularity of information which is phenomenological, methodological, recursively formal, linguistic, biologic, etc. This property is spontaneous, comprehensive, informationally generative, parallel, serial, sequential, etc. Informational recurrence is information by itself.

THE CONSEQUENCE OF INFORMATIONAL RECURRENCE. The word 'recurrence' has the meaning of the coming back of information into existence, embedding, Informing, and Counter-Informing, in a spontaneous circle which is understood as Informing in the broadest meaning. An example of static, constructive informational recurrence is a mathematical recursive formula, by which it is possible through the usage of formal linguistical means to obtain a new formal or numerical (logical) result recursively. Informational recurrence is dynamic, so that, for instance, the mentioned recursive formula is changed and developed by the process of recurrence. Besides this, informational recurrence has to be understood as an informationally parallel and/or informationally serial (sequential) recurrence. Serial recurrence seems to be obvious and imaginable, although the possibility of arising of information within serially structured information has to be taken into account. However, parallel recurrence has to consider the possibilities of arising of parallel information within already existing parallel and serial information. In this respect, informational recurrence is not limited in any way, so that information and its Informing can spontaneously arise in an informationally circular way. Finally, it has to be mentioned that information also concerns informational recurrence as information circularly, counter-informationally, and with inner informational comprehension, which is a property of information itself.

AN EXAMPLE OF A GENERAL INFORMATIONAL RECURRENCE. Informational recurrence, in which the parallelness and/or serialness of informational forms and/or informational processes are not explicitly determined, can be demonstrated by the following informational formula:

```
                informational_recurrence
                      means_that
            information arises_from information
              as counter-information after_which
counter-information is_embedded_in information
                      and_in_this_way
                arising_of_and_embedding_of
                information is spontaneous;
```

In this formula the operator after_which is clearly serial in concern to the previous part of the formula. The operator arising_of_and_embedding_of represents a parallel structured operator, by which arising and embedding of information have to be understood as parallel informational processes. Certainly, the explicitness of parallelness and serialness can be expressed by the introduction of separate operations (e. g. is_parallel_to, is_in_series_with, etc.).

THE PRINCIPLE OF PARALLELNESS OF INFORMATION. Informational parallelness is a circularity of information which is topological, spatial, alternative, multiserial, parallel-generative, parallel-comprehensive, etc. Informational parallelness is information by itself.

THE CONSEQUENCE OF INFORMATIONAL PARALLELNESS. The existence and arising of parallel information needs the possibility of parallel appearance of informational forms and informational processes. In this case the word 'possibility' represents the notion of parallel structure which enables parallel informational appearance. This structural parallelness is in some way topological, spatial, alternative, spontaneous, etc. The existence and arising of a parallel informational structure qua informational substance is the necessary condition for informational parallelness as stated in the previous principle.

A population of living beings is a good example of informational substance, in which parallel information appears in spontaneous and various ways. This parallel information influences beings in their interactions, so that these beings are informationally interwoven to some extent. This example shows how parallel information as an informational unity of population is unforeseeable in its existence and arising, depending from the circumstances of population's environment or from the environmental information. The parallelness of information appears when an existing informational pathway is split in two or more pathways.

AN EXAMPLE OF TWO PARALLEL PROCESSES. Let us introduce two distinct informational entities, label them ip1 and ip2 and let them represent two parallel processes. These two processes are informing mutually independent; however, counter-information arising from a process is embedded in itself as well as in the other process. Informing of the process ip1 and ip2 depends solely on their own counter-information. In this way, information is arising dependently on both of processes, Informing of each process is arising only within each respective process. We can construct the following formula:

```
                ip1 is_parallel_to ip2
                    in_the_way_where
                counter-information of i1
                      influences
  information of i1 and i2 and Informing of i1
                  and_parallel_to_this
                counter-information of i2
                      influences
  information of i1 and i2 and Informing of i2;
```

So far, this formula, concerning the previous text, is clearly understood. The operator 'influences' has in fact the more precise meaning of 'is_embedded_into'.

THE PRINCIPLE OF SERIALNESS OF INFORMATION. Informational serialness (sequentialness) is a temporal, looping, unidirectional, memorial, spatially closed, etc., spontaneously arising circularity of information. Informational serialness is information by itself.

THE CONSEQUENCE OF INFORMATIONAL SERIALNESS. The existence and arising of serial information needs the possibility of serial appearance, of

interpolating, of inserting information between information and of serial reconfiguring of serial information. Although informational serialness is temporally structural, it can also be understood in a parallel way. Serialness of information points to the fact that temporality or time has to be comprehended as an informational difference appearing in the relation between information and counter-information. There is no clear distinction between informational serialness and informational parallelness, but in some cases this distinction may be useful.

AN EXAMPLE OF SERIAL AND PARALLEL QUESTIONING. Consider three serial processes which are labeled by question, questioning, and interrogating. The question process is producing questions as its counter-information, and this counter-information is embedded into the information of the question and questioning process and into their Informings. The questioning process is producing questioning as its counter-information, and this counter-information is embedded into the information of the questioning and interrogating process and into their Informings. The interrogating process is producing interrogating as its counter-information, and this counter-information is embedded into the information of the interrogating, question, and questioning process and into their Informings. This process scheme is circular concerning question and questioning, questioning and interrogating, and interrogating and question. We can state the following formula:

serial_process_of_question_questioning_
_interrogating
is
question informs and
counter-information_of_question
is_embedded_in
information of question, questioning
and_in Informing of question, questioning
afterward
questioning informs and
counter-information_of_questioning
is_embedded_in
information of questioning, interrogating
and_in Informing of questioning, interrogating
afterward
interrogating informs and
counter-information_of_interrogating
is_embedded_in
information of interrogating, question
and_in Informing of interrogating, question
afterward
serial_process_of_question_questioning_
_interrogating
is repeated;

A parallel informational formula, where question, questioning, and interrogating are parallel processes, is the following:

parallel_process_of_question_questioning_
_interrogating
is
question, questioning, interrogating
is_informing and
counter-information of
question, questioning, interrogating
is_embedded_in
information and Informing of
question, questioning, interrogating;

The circularity of this formula is expressed by the postfix operator is_informing.

4. Informational Form and Informational Process

If information is understood as being-in-the-form and being-in-the-process, then it is reasonable to introduce the notions of informational form and informational process. Being-in-the-form concerns any imaginable form as information. Being-in-the-process concerns any imaginable phenomenon or phenomenal complexity as information. This point of view is acceptable for living beings in which information is coming into existence. Even living beings themselves can be comprehended as informational forms and informational processes, which are informing themselves and which are being informed by outside informational forms and informational processes concerning them. On the basis of this consideration the following principle can be accepted:

THE PRINCIPLE OF INFORMATIONAL FORM AND INFORMATIONAL PROCESS. The entire living and non-living phenomenology can be considered, perceived, observed, investigated, recognized, understood, etc. as a phenomenology of informational forms and informational processes. Forms and processes of the physical and psychical world inform and are mutually informed, that is, they disturb each other passively and actively as well as themselves. Any phenomenology is a phenomenology only to the extent by which it is an informational phenomenology, by which phenomenal forms and phenomenal processes are perceived as information and Informing of information.

This principle offers a particular, informational orientation and may begin a new tradition, if everything is understood as informational. This is the principle of informationism or informism as determined in (IDI). If everything is understood as information, to be an informational form or informational process, then this realm of understanding is called informationism. Informism is nothing else then the thinking and understanding of the entire phenomenology through Informing or by means of Informing, through the way in which information informs the living and non-living.

THE CONSEQUENCE OF INFORMATIONAL FORM. An informational form constitutes information which it gives through its Informing or receives from an informational source. Although, informational form is a static phenomenon, it can cause Informing in the receptor. In this way, an informational form is informing and a sharp distinction between informational form and informational process is not possible. Informing of an informational form is the possibility of its observation, investigation, and cognition, where an informational form is producing its counter-information. In this respect, an informational form is a very general informational principle and is information by itself.

THE CONSEQUENCE OF INFORMATIONAL PROCESS. An informational process is a dynamic informational image of a phenomenon, it is a process within information, which actively and passively informs information within itself as a process. An informational process changes itself through Self-Informing and through Informing of other informational processes. It generates counter-information by itself and takes this counter-information and to-itself-outward information into its own informational consideration. An informational process is generating informational forms and informational processes inside and outside of

itself. Every informational process is information by itself.

5. Structuring and Organizing Information

What is the fundamental distinction between the structure and organization of information? Is a rigorous separation between them possible at all?

THE PRINCIPLE OF INFORMATIONAL STRUCTURE AND INFORMATIONAL ORGANIZATION. Informational structure is a constitution of information, that is, a constitution of informational forms and informational processes that are composed as information. These forms and processes are informational components. The informational relations among informational components which determine a composite information constitute informational organization. In terms of informational epistemology, informational structure is closer to the form, whereas informational organization is closer to the process. Within information, informational forms and informational processes are informationally interwoven components. Informational components integrate information. Informational structure and informational organization are information by themselves.

THE CONSEQUENCE OF INFORMATIONAL STRUCTURE AND ORGANIZATION. Informational structure is information concerning type and existence of informational components. Informational organization is information concerning type and existence of informational relations, dependencies, necessities, etc. among information-constitutive components. From this viewpoint, information is an informationally structured and organized phenomenon. Informational structure and informational organization inform (i. e., arise, embed, are embedded, and counter-inform) in a circular manner (i. e., recurrently, parallel, and serially).

There is an essential difference between informational structure and informational organization on the one side and the usual structure and organization on the other. Structure and organization of information arise as information. Therefore, information, as a unity regarding its structure and organization, is changing and coming into existence with new informational components, relations, etc. which are embedded in existing information. Existing information, which informs as a given informational background, is changing and arising. Evidently, the consequence of informational structuring and organizing is the openness or non-closure of information as unity and as autonomy.

EXAMPLES OF CULTURAL FORMS. Cultural forms (e. g. philosophy, ethics, ideology, science, art, etc.) qua information are ripe examples of particular informational structures and informational organizations. The structural and organizational particularity of a cultural form is called informational Enframing. Such an Enframing, regularly embedded in information, exists outside of a particular cultural form. In general, an informational Enframing, representing a cultural form, is going to be embedded in a broader informational realm, which forms the background of the given Enframing. For example, mathematics as a discipline is enframed in a mathematical structure and organization to preserve its mathematical rigorousness, i. e., its mathematical Enframing. This Enframing is embedded in a broader informational realm, in several other scientific disciplines, in philosophy, language, etc., where mathematics has its roots, application, influence, etc.

The structure and organization of information constitute informational Enframing, which is informationally embedded in a broader informational context. In this respect, no scientific discipline or cultural form can be purely scientific or purely cultural, that is, closed purely in itself as a scientific discipline or a cultural form. Informational Enframing is the principle of informational structuring and organizing and is information by itself.

6. Intelligence as Information

> ... The essence of intelligence is to act appropriately when there is no simple pre-definition of the problem or the space of states in which to search for a solution. ... Heidegger demonstrates that the essence of our intelligence is in our thrownness, not our reflection. Similarly, Maturana shows that biological cognitive systems do not operate by manipulating representations of an external world.
>
> (Winograd, Flores, UCC, 98, 99)

In general, intelligence as information is nothing more than an informational complexity, a destined, intentional, and goal-oriented composition of information. In this respect, intelligence is an informational product produced by divergent, higher informational functions of a living information system. Information constructs intelligence by the intention that this intelligence can serve as a proper information for distinct situations of a being's being, for solving real and artificial problems.

THE PRINCIPLE OF INTELLIGENCE. Intelligence, in which embedded information is intentional; informationally enframed, grounded information (i. e., in a consequent structure and organization) and in a specific domain (discipline, realm), is arising to the informational surface for specific problem solving (behaving, thinking, acting). Intelligence comprehends (to observe, investigate, recognize) only a specific problem domain and it is only in this domain that it influences behavior and action intelligently. Intelligence is specialized information, which is problematically enframed, closely concerns a problem domain, and strictly comprehends in the metadomain of a problem domain. For this purpose intelligence needs other information, comprehension (processing) of other information, instruction, knowledge, and a characteristic informational expertise.

THE CONSEQUENCE OF INTELLIGENCE AS INFORMATION. Intelligence is information, which is narrowed into a specific problem domain, with its own solving performance, experience, and characteristic methodological blindness. Intelligence knows its own systematical approach, its way of recurrent descent. In intelligence's comprehensive metadomain there are specific instructions, knowledge, expertise, methodologies, concept attainment, and intelligence's own specialized tools and procedures of problem solving. Intelligence always needs its characteristic informational substance out of which it arises and into which it can be embedded. Arising of intelligence depends upon the intelligent environment, upon the environing world in which it is dwelling and from which it can grow.

7. Informational Machine and Informational Program

> ... Even if a computer program consistently wins games of chess, Winograd and Flores would say that this is not intelligence.
>
> (Clancey, AI, p.243-244)

The notion of an informational machine and an informational program concerns the contemporary philosophy, science, and technology. The question of future machine and program development must consequently be founded upon the informational point of view, that is, from the point of view of philosophical and implementational possibilities of machines and programs as informational entities.

THE PRINCIPLE OF AN INFORMATIONAL MACHINE. A machine is informational if its structure (substance, form, structural components) and its organization (machine connectedness, relations, processing) are informational. The structure of a machine concerns the machine's architecture (physical, biological constitution). The organization of a machine concerns its functioning (mind, behavior) and its flexibility (controllability, progammability) under various inward and outward conditions. Architecture of an informational machine is dynamic (brain-like), is dynamically controlled, interchangeable, arising, and depends upon the machine's environing world. An informational machine can be governed by informational programs to achieve the required informational characteristics and properties. The informational machine performs as information.

THE CONSEQUENCE OF THE PRINCIPLE OF INFORMATIONAL MACHINE. Because an informational machine possesses informational attributes (Informing, embedding, arising, Counter-Informing in an informationally circular manner), its performing is informational. This performing can be achieved through its informational architecture and through its informational programmability. A dynamic architecture (architectural switching without architectural arising) already represents an informationally bounded machine. This weakness in informational dynamics can be eliminated by informational programs. Hence, an informational machine possessing the full (or even partial) informational power is currently not technologically available.

Several new research and development projects have to be initiated in order to discover informational architecture components which would have a richer informational variability, especially parallelness and the feasibility to be used in dynamic architectures. These architectures could temporarily change and can even arise or develop as a consequence of their changing and arising environing conditions.

THE PRINCIPLE OF AN INFORMATIONAL PROGRAM. An informational program would simply be information which spontaneously informs, embeds, arises, and counter-informs in an informationally circular way within an informational machine. An informational program informs itself and other informational programs and is used and embedded in an informational machine for production of information (e. g., intelligence, dedicated informational functions, etc.).

Evidently, there is an essential difference between a computer program and an informational program. The former is algorithmic (mathematical, procedural, informationally static), whereas the later is informational (intelligent, informationally dynamic). As a rule, a computer program has a stable, non-variable program structure and program organization. Its definition can not be changed dynamically during its execution by the parallel execution of itself and other programs, data, etc.

THE CONSEQUENCE OF THE PRINCIPLE OF AN INFORMATIONAL PROGRAM. An informational program performs as information. In this respect, such a program is also an informational object which can be informationally changed during its processing. A typical computer program is always performing as a subject, by which non-program objects can be changed and can arise as results of its performing. In principle, the request to informatize a program concerns essentially different programming tools as compared to those that are still used today.

AN EXAMPLE OF A TECHNOLOGICAL INFORMATIONAL MACHINE. A technological (artificial) informational machine mainly concerns an architecture, which is completely different in comparison to today's usual computer architecture and its components. The so-called brain-like architecture is only a typical representative of this new orientation. What does the brain-like architecture mean? Roughly, components of a brain-like architecture resemble neurons and functional units (regions, nuclei, areas, cortices) of neurons (neuronal populations). A neuron is a nerve cell (a specialized processor) with all its various and complex informational processes. In the living brain no two neurons (basic cell processors) are identical. Signals (information) to a neuron are entering from other neurons via synapses (synaptic processors). Neurons can appear and disappear (functionally as well as physically). Synaptical connections among neurons can arise in dependence from various inner (structural) and outer (processing) requirements. A brain-like architecture is developing through its life under the pressure and influence of inner and outer circumstances. It is arising dynamically and performing as information on structural, substantial (technological) level.

AN EXAMPLE OF AN INFORMATIONAL PROGRAM. An example of informational program is a cortical (brain-like) function. Such a program is not only performing as information, it is also influencing its subsequent substance (architecture) in which it is developing (in which it is processing). In this respect, it can be understood that the mind (as informational program) is influencing the development of the brain (as informational architecture with its components) and that such an arising architecture is offering new possibilities for the arising of mind.

8. Living Examples of Information

In the previous two examples a living informational machine (brain) and a living informational program (mind) have been described. According to Maturana and Varela (AAC), the theoretical treatise of living organisms can be understood as autopoiesis.

THE PRINCIPLE OF A LIVING INFORMATIONAL MACHINE. A living informational machine represents a subclass of possible (imaginable) informational machines. The living informational machine is constituted by its biological structure and organization, by its autopoiesis (Maturana, Varela, AAC). Such a

machine is organized as an informational network of informational forms and processes for production, transformation, and destruction of its own information. A living informational machine governs the life of the machine, preserves its own life, and produces itself and its information.

Obviously, as a product of their functioning, living informational machines have something which concerns themselves, their own identity, unity, and life processing. Functioning which does not concern the machine itself is characteristic for artificial, allopoietic machines. These machines do not develop and preserve themselves, however, they produce functions for other purposes (e. g., computers, mechanical machines).

THE PRINCIPLE OF LIVING INFORMATIONAL PROCESSING. Living informational processing is the biological phenomenology of autopoietic informational systems in the living world to the extent where this processing depends in one way or another on the autopoiesis of one or more autopoietic informational units. The domain of all interactions in which an autopoietic informational system can enter is its cognitive domain. Within this informational domain the autopoietic identity cannot be lost.

AN EXAMPLE OF LIVING INFORMATIONAL MACHINES AND LIVING INFORMATIONAL PROCESSING. Evident examples of a living informational machines are molecules of life, cells, cells populations (organisms), cortical nuclei, cortices, brain, and a living being as a whole. Liying informational processes in these machines are, for instance, synthesis of proteins, immunity, evolutionary learning, variation, selection, replication, mind, etc.

9. Formalization of Informational Principles

In the examples of this essay several informational formulae have been treated. It has been shown how informational forms and informational processes can be formalized by linguistic means. Information is an informational conception and it could be formalized sufficiently only by informational means. Information itself appears to be a kind of informational system in itself. Within this system it is arising and embedding circularly and, in this respect, it has an informational structure, organization, forms and processing. Information is being thrown into an informational environment from some *informational source, from where it arose,* and from this point on it is developing, arising, growing, vanishing, etc. like a specific system, which is related to itself and to its information environment. Through the appearance of information its system is coming into existence.

The question which now arises is how an information-related, information-characteristic, or information-substantial system can be formalized. It is more or less evident that this formalization cannot be a mathematical, algorithmical, or logically formal presentation. There do not exist means, a formal apparatus, or a mathematical formalism which could be used for a soft and satisfactory *informational formalization.*

In the previous examples a formal symbolism for informational formalization was not introduced. The trial was based on a level of natural language. The aim of this formalization was to present informational principles and operations in a more comprehensible way of linguistically (verbally) exposed conceptions. The formalized examples could be followed quite easily, systematically, clearly, and recurrently. Generally, the form of formalization was always an arbitrary sequence of informational operands and informational operators, irrespective of parallelness, serialness, or circularity of operands and operators. It was shown that a *sequence can also be used for explicating* parallel operations.

References

(AI) W. J. Clancey (Book Review). Artificial Intelligence 31 (1987) 232-250.

(SCC) P. J. Denning: The Science of Computing. American Scientist 75 (1987), No. 2, 130-132.

(ESD) S. J. Gould: Ever Since Darwin. Norton. New York (1977).

(CCH) R. V. Jensen: Classical Chaos. American Scientist 75 (1987), No. 2, 168-181.

(AAC) H. R. Maturana and F. J. Varela: Autopoiesis and Cognition. The Realization of the Living. D. Reidel Publ. Co. Dodrecht, Holland (1980).

(UCC) T. Winograd, F. Flores: Understanding Computers and Cognition: A New Foundation for Design. Ablex Publ. Corp., Norwood, NJ (1986).

(OUC) T. Winograd: On Understanding Computers and Cognition: A New Foundation for Design. A response to the reviews. Artificial Intelligence 31 (1987) 250-261.

(OWI) A. P. Zeleznikar: On the Way to Information. Informatica 11 (1987), No. 1, pp. 4-18.

(IDI) A. P. Zeleznikar: Information Determinations I. Informatica 11 (1987), No. 2., pp. 3-17.

DECISION SUPPORT SYSTEMS: TOOLS, EXPECTATIONS AND REALITIES

Janez Grad and Milton A. Jenkins*
Ekonomska fakulteta Borisa Kidriča, Ljubljana, Yugoslavia
* School of Business, Indiana University, Bloomington, Indiana, USA

UDK 519.86

In this paper we describe and analyze Decision Support Systems (DSS) as a class of Information Systems. We first state a short definition of DSS, explain its relation to Management Information System, and the objectives of DSS. Afterwards, our major discussions are devoted to the following issues respectively: (i) The iterative design for DSS (prototyping approach) as part of the broader aspect of DSS building, where we consider cost and benefit impacts and the possible problems and bottle-necks of this approach. (ii) The current computer and information technology that supports DSS, e.g.: on-line computer systems with the necessary software support for interactive use, software packages for computer graphics, statistics, operations research, financial planning, etc., quick hit DSS programs, fourth generation languages, database management systems and so on. The advantages, disadvantages, and limitations in practice are discussed. (iii) The current status and application of DSS for marketing analysis, sales forecasting, financial planning, transportation, human resources management, use of graphics in decision making, etc. (iv) The future trends in DSS. We give a broad overview of the expected development and use of DSS, in particular the further impact of technology and the necessary changes in organizations and decision processes.

Some methodologies and program packages that were listed above were developed in the research programs carried out at the Graduate School of Business, Indiana University, Bloomington, Indiana, USA, and Ekonomska fakulteta Borisa Kidriča, Ljubljana, Yugoslavia.

1. INTRODUCTION

In this paper we describe and analyze Decision Support Systems (DSS) as a class of Information Systems (IS) that support decision-making activities of managers and others who are involved in decision-making procedures and processes, the technology of DSS and its application, and the trends and future directions of DSS and supporting technologies.

We first discuss the relations between Electronic Data Processing (EDP), Management Information Systems (MIS) and DSS. Different views about this issue are briefly stated and explained, taking into consideration the personalities of various authors. The practitioners and the theoreticians frequently view the problems differently. We then concentrate our attention on the variety of different technologies and their applications in DSS. We detail more the most important technologies in DSS, emphasizing those on which some reasonable amount of research has been conducted within the high schools the authors belong to. For instance, data base management, analytical methods, computer graphics, spreadsheets, prototyping and fourth-generation languages (4GLs). In conclusion we summarize the general views and believes on the future trends and developments in DSS and DSS technology. Short-term trends up to 1990, and long-term trends, after 1990, are briefly discussed. We believe these forthcoming technological and organizational changes in DSS will have a significant impact on future high school curriculum and pedagogy in the MIS area.

2. DECISION SUPPORT SYSTEMS VERSUS MANAGEMENT INFORMATION SYSTEMS AND THE OBJECTIVES OF DECISION SUPPORT SYSTEMS

Within the steady advancement of computer-based IS in organizations a new stage has been reached where the term DSS has been introduced. Different explanations of DSS have been provided. Some view DSS as another step in the natural evolutionary advancement of information technology and its use in the organizational context, following EDP and MIS. Others view DSS as an important subject of MIS or just a type of system that has been developed and used for several years already but has only recently been uniquely defined. Some claim that the term has been introduced merely to attract people but it does not define anything particular new in the field of computer-based IS. We discuss briefly these issues in the following lines, state some definitions on DSS and in this way define the subject which we present and analyze in this paper.

Many vague, either restrictive or very broad, definitions of DSS were put forward in the 1970s. They didn't help to clarify which of the above stated explanations were appropriate

and which were not. The first definition of DSS as an interactive computer-based system that helps decision makers utilize data and models to solve unstructured problems was later extended to all the systems that contribute to decision making. Some examples of complex systems were also examined. More promising than the definitional approach or the example approach was the "characteristics" approach of DSS [2, 14] that associated with DSS the following characteristics:

- they are aimed at the less well structured, underspecified problems that upper-level managers face
- they combine the use of models of analytic techniques with traditional data access and retrieval functions
- they focus on features that make them easy to use by noncomputer people in an interactive mode
- they emphasize flexibility and adaptability to accommodate changes in the environment and the decision-making approach of the user

In short, DSS should support managers in their decision-making activities. The ideas and research resulted in the form of programmed packages for building DSS, first used on mainframes and later on personal computers.

Based on the previous development and research, Sprague and Carlson [21] gave the following definition of DSS:

"Computer-based systems that help decision makers confront ill-structured problems through direct interaction with data and analysis models".

According to them a good DSS should have the following three capabilities:

- it should be easy to use and should support the interaction with nontechnical users
- it should have access to a wide variety of data, and
- it should provide analysis and modeling in a variety of ways

In order to make a clear distinction between the terms MIS and DSS we must go a bit further in our analysis. There are two general views on what MIS and DSS should be. The first view, the theoretical view, is the view of academicians who are concerned about and who develop theoretical backgrounds of DSS. The second, so called connotational, view is the practitioners' view, who make conclusions and definitions on the basis of their experiences in creating and using some particular MIS and DSS. It is largely the practitioners who support the view that MIS is an advancement of EDP and that DSS is a further advancement of MIS.

Connotational view on EDP-MIS-DSS differentiates the three terms on the following basic characteristics:

- EDP emphasizes optimum data processing with the outputs aimed and used at the operational level
- MIS emphasizes integrated information acquisition environment based on DBMS with the outputs aimed at the tactical level
- DSS emphasizes user decision-making real-time information acquisition environment aimed at top managers and executive decision makers

The view is partially supported by case studies, but it is nevertheless inappropriate as far as the future development of DSS is concerned. Decision-making is not the exclusive domain of the top level management. Decision-making must be distributed across all three functional levels.

The theoretical view proceeds from and is based on the following objectives of the IS function:

- improving performance in order to get the right information to the right person at the right time
- users involved in IS are knowledge workers, such as managers, professionals, and other employees, who also are responsible for further development of information technology
- the paradigm of IS is a goal-seeking organization

This view places DSS among other major technology subsystems of IS which are interacting with each other and other application systems and which are supporting users on all vertical levels of management, not only at the very top level. Other major technology subsystems are (i) the system to support communication needs and (ii) the structured reporting system. This standpoint is defended by a general merging of information technology, operations research, statistics and management science approaches in the form of interactive modeling [21]. DSS is accepted as evolutionary advancement in the systems dimension of a three-dimensional IS model which evolved from the two-dimensional model of MIS. DSS requires new strategies in design of the information systems technology and its interactive usage, from those used in MIS.

Hackathorn and Keen [10] interpret DSS in term of the number of people that participate in the decision-making process as

- independent decision-making where a decision-maker makes decisions. This approach requires personal support
- sequential interdependent decision-making where a decision-maker makes part of a decision, which is then passed on to someone else. The approach requires organizational support
- pooled interdependent decision-making where several decision-makers negotiate and interact in order to make a decision. This requires group support

Experience shows that new approaches and technologies in IS usually promise more than they deliver. Frequently the real contribution is something quite different from what was expected at the very beginning. MIS and office automation diverted from their promises [22]and became well-structured reporting systems (instead of electronic nervous system for organizations) and word processing on personal computers (instead of a paperless office). Nowadays users still believe in the promise of DSS as defined above.

3. DSS CONCEPT, TECHNOLOGY OF DSS, AND APPLICATIONS OF DSS

Traditional data processing systems (such as payroll, inventory, airline and similar reservation systems) since the early 1960s have typically taken the form of predefined reports. Infomation is produced by either aggregating or disaggregating data within the system. The

information is static and rigid and any question that had not been included in the process would demand a difficult and time-consuming procedure, including new programs and changes to the data structures. Traditional EDP and MIS are not "user-friendly" (require programmers), their technology, which the programmers use, is rigid, hard to change and demands a lot of programmer time in order to produce results. Questions to the system must be predefined and not some unusual requests from the high-level decision makers.

DSS commonly copes with problems that are not structured, i.e., no procedures for their solution are known. Such problems-decisions, are for example: planning the amount of organizational expenditure, deciding whether to introduce a new product or program, such as a new airline, or a new technology procedure. Decision-making involves multiple criteria and results in a number of trade-offs which are analyzed and modified iteratively by the decision-maker possessing great experience [7]. It requires an interactive computer system with all the necessary support software and a considerable amount of data which the decision-maker uses while exploring the problem. The system must support the decision-maker by suggesting solutions and the possible consequences of accepting a particular solution. Within the subsequent steps of the iterative process the decision-maker may back-track, modify, refine and introduce more sophistication into the solution several times until, he finds a satisfactory final solution. Thus, in this way, a DSS becomes a tool for building a model or creating a solution of the future state of the business, based upon sets of assumptions and relationships supplied by managers and other users.

DSSs need subsystems with data and algorithms that a decision-maker can use. These subsystems can sometimes act independently but they can also be inte rated. Examples of such application are: (1) to retrieve a simple data item while processing an order for a particular commodity, (2) to generate a report of all the firm's foreign customers in the last year, (3) to use mathematical programming algorithm in order to perform allocation planning, (4) to perform a formal statistical analysis to find the correlation between different variables, (5) the use of models, where an expert DSS system has been created, based on the past many years' experiences and decision-making procedures of some expert. The present decision-maker gets help by using this model when he has to make a decision.

The DSS approach is a user-friendly approach. Users acquire information from a DSS without the help from a programmer. They do this by using a query language and/or a request generator. Instead of procedural languages, like COBOL, FORTRAN, etc., DSSs use the fourth-generation languages. In 4GLs one statement is equivalent of many statements in a procedural language. These languages also use system prompts and help commands in order to make them easier to use and understand.

In the following paragraphs we discuss the technology necessary for implementing the DSS concept. We restrict ourselves mainly to the computer software and techniques used in problem-solving.

One of the most important methodologies is a DATA BASE MANAGEMENT. The DSS user (manager) frequently retrieves data items from a DB randomly, produces reports from a DBMS or creates and manipulates more complex logical data structures for a system which allows him to produce models involving data of certain properties. Present DBMS are still part of the software, although in the future they will probably become part of hardware or firmware. They are also available on microcomputers and personal computers, not only on large and powerful mainframes. Software products with imbeded DBMs such as FOCUS, RAMIS II, NOMAD2, EXPRESS, GADS, EIS, REGIS, GMIS, etc. are available on the market. They are expensive to buy (but prices are hopefully declining) and they require large amounts of computing resources [7, 21, 22]. Therefore, initially many of these systems were being used in independent computing companies or in computers manufacturers Efficient use of this technology is based largely on the adequate supporting documentation, such as:

- data flow diagrams, which present a graphic model of processing, the storage of data and the movement of data
- data dictionary, which contains the terms and their definitons
- process descriptions, where each process bubble must be described in sufficient detail [18]

When building the DB management part of DSS we must choose one or more basic data structures, i.e., a method of representing and retrieving data in a computer. In addition to the four well-known data models used in MIS -- the record model (flat file), the hierarchic model, the network model, and the relational model, one more model, the rule model, is being used in DSS environment [21]. This model is common in artificial intelligence systems and in so-called "knowledge-based" DSS. It specifies production rules and enables making inferences based on the data. The rule model describes data by a set of rules, i.e., a set of data definitions. The choice of model should not be based on the representation of the data, but on the operations and integrity constraints.

Among the ANALYTICAL METHODS that DSS needs for analysis and modeling are statistical procedures, data projection or simulation and optimizing models. Statistical packages, such as SPSS and IDA are being used in many universities and firms all around the world. The Interactive Financial Planning System (IFPS) is an example of the financial planning modelling languages for data projection and simulation. It can be used on VAX and some other minicomputers, and mainframes. IFPS has been taught and used in the Graduate School of Business, Indiana University, Bloomington, Indiana, USA as an efficient programming tool for several years, and it has been widely adopted in the USA, in both industry and universities. Its impetus came from the desire to model risk in a way which could easily be understood by executives. IFPS has a self-contained non-procedural language which is easy to understand and use [8]. The logic and output of IFPS resemble those of spreadsheet packages. A number of third-party computer software packages compatible with IFPS are available; for instance: SENTRY for data entry in a form compatible with IFPS; DATASPAN for converting data bases in arbitrary format to a form usable by IFPS; GRAPHICS for presenting IFPS results on color graphics displays; and OPTIMUM for finding optimal solutions of IFPS models by linear, nonlinear, or integer programming. Optimizing models are usually based upon mathematical programming algorithms. A group of researchers at Ekonomska fakulteta Borisa Kidriča, Ljubljana, Yugoslavia has pursued this field of research for many years,

developed several program procedures in Operations Research (linear and dynamic programming) and published numerous research papers [4, 5, 19]. Part of this software development was supported by Intertrade, Ljubljana, the IBM representative in Yugoslavia, the Yugoslav computer manufacturers firm Iskra-Delta, Ljubljana and the software house Iskra-CAOP, Ljubljana.

INTERACTIVE COMPUTER CAPABILITIES must be available in DSS environment. The following approaches of DSS computer implementation are now possible:

- DSS software on a large-scale general computer which users access from terminals in an interactive mode
- DSS software on a dedicated minicomputer, where users are involved in different DSS applications simultaneously through their terminals. Here the special DSS software does not slow down other jobs being run on the mainframe. A disadvantage of this approach is that the required central data must often be transmitted to the minicomputer through magnetic tape, etc. which makes the system slow
- the use of personal computers to host DSS of smaller size. This approach is popular because of easy accessibility for the user. Serious disadvantages are far less powerful software than is available on mainframes and minicomputers and relatively small secondary storage capacity. Also the abstraction and loading of central data on microcomputers are even more awkward than in the case of the minicomputers

Some DSS (now being labeled Executive Information Systems (EIS)) help decision-makers by employing built-in EXPERT SYSTEMS -- expert's decision-making procedures. To build such a system, the expert's decision-making process is studied and a computer program is written that behaves similarly to the expert. Expert systems have been used in medical diagnosis, oil exploration, computer chip design, and in various business applications (auditing, making commercial loans, financial planning, etc.).

An especially important technology for business problem-solving and decision-making is COMPUTER GRAPHICS. It helps managers to acquire visual representations of data, relationships and summaries for information activities are not based on predefined processes or procedures. Graphics allows the users to view or search the data in new and creative ways in the context of their particular problems or goals. Several graphics forms can be generated by computers, such as texts, time series charts, bar charts, motion graphics, scatter diagrams, maps, hierarchy charts, sequence charts, etc. They can be used in DSS in many different ways such as: reports, presentations, management tracking of performance, analysis, planning and scheduling, command and control, for design, engineering and prodiction drawings. Graphics can also be used in computer-aided design, computer-aided manufacturing, teleconferencing and videotex systems. The benefits of computer graphics over manual graphics are in costs and time. Formats, scales and colors can be tested in order to obtain the best comprehension of the information. Objections to computer graphics are that high resolution graphics are still very expensive; sometimes the graphs are of low quality, and require skilled and experienced users who can produce good graphics. Some of these objections will become irrelevant with the new computer graphics products.

The Operations and Systems Management department at the Graduate School of Business, Indiana University, Bloomington, Indiana has made in the last decade extensive research on the effects of different forms of computer graphics presentations, their complexity and color of presentation on the human decision-maker, see for example [9]. This line of research is a part of a broader program of research, called PRIMIS - Program of Research for Investigating MIS, which focuses on the user-system interface of DSS. A theory of graphics information presentation has been formalized: Performance with a given information presentation is a function of question difficulty, information complexity, the form of presentation and color.

Another widely accepted DSS technology are SPREADSHEETS. They are self-documenting systems with explanatory internal documentation and prompts that help the user to proceed his dialog with the computer from one step of the problem solving procedure to another. Their main advantages are that the user gets the data in a table form on video screen and the relationships between data series in a form of report. The user may test the impact of some particular data item or group of data items and/or relations among them on the model's output. He can do this by ineractively entering different values for data items, temporarily changing the algorithm and analyzing the computed results. This "what if" capability is present in many spreadsheet packages on the market. These spreadsheet packages are aimed at problem-solving and model-development in fields of financial planning (amortization, depreciation, lease-versus-buy, discounted cash flows and net present value), real-estate investments (finansing alternatives,impact on taxes, payoffs, cash flows), business record-keeping and accounting, budgeting and statistics. Despite their popularity, these spreadsheets have many inherent weaknesses, such as the difficulty of specifying all data requirements a priori, data-model dependence, limitation of the relations that represent the model and model's complexity by the spreadsheet's table format, user's session cannot be recorded and little flexibility in report writing features. It is estimated that 20 to 30 percent of the users will become dissatisfied with spreadsheets and will ask for more powerful tools [22]. Spreadsheets appear to be most useful for smaller problems. The possible solution to the existing variety of many different spreadsheet systems which the users have to learn, would be integrated packages that will combine spreadsheets, word processing, data management, graphics, data communications, and other resources. SYMPHONY is a (not too successful) example of this trend in the software market. SYMPHONY expands the capabilities of LOTUS 1, 2, 3.

QUICK HIT DSS [16] is a term that explains a special procedure used more and more in DSS. It stands for a rather simple DSS prototype which the decision-maker creates and processes before he decides whether to build a full DSS or not. Three types of quick hit DSS include:

- reporting DSS
- short analysis programs
- DSS generators

Reporting DSS is a very frequently used form of decision support which includes simple data manipulations (selecting, summarizing, and listing data from files, some other arithmetic operations on these data, presentation of trends and variances, by means of computer graphics) in order to meet some information needs of decision-maker.

Short Analysis Programs are used for analyzing data. They need small amount of data and are usually written by decision-makers themselves

in BASIC or some other high-level programming language. Functions that help the decision-maker to make decisions include projection of costs, income, and profits; allocation of fixed costs among products; project management; graphing of some activity output figures, etc.... Examples of short analysis programs are studied in [1, 6, 16].

Decision Support System Generators are products which include languages, interfaces, and other facilities that help to frame up specific DSS. DSS generator can be used to build more specific DSS within a class of applications. In recent years users are generally interested in DSS generators and fourth-generation languages and not much in the other two types of quick hit DSS.

These quick hit approaches are not appropriate in areas such as forecasting or allocation of resources where a deep understanding of sophisticated methods, techniques and the whole application area is necessary and the final models cannot be replaced by some approximations of them in order to make some starting decision [3].

PROTOTYPING has been defined in many different ways: as a philosophy, a methodology and a procedure. Within each of these classes of definition are further differences. For example, as a methodology for the development of information systems two definitions are very evident: the "rapid prototyping" approach from computer science and the "prototyping methodology (PM)" from MIS. The PM [12] is most appropriate in discussing DSS and is broad enough to encompass the various more limited definitions found in the DSS literature. Under the PM the process for building an operational prototype is described and this operational prototype may be used in various ways, e.g., stand alone, with life-scale methodologies, as pilots or prototypes and as "throwaway" programs. Senn describes prototyping as one of the seven activities within the system development life cycle[20]: preliminary investigation, determination of requirements, development of prototype system, design of system, development of software, systems testing, and implementation. This approach is used when we cannot define all the features of the system in advance, due to the lack of experience or information, or when we face high-cost and high-risk situations. In such cases an inexpensive small-scale version of the software is prepared in order to provide some preliminary information about the environment in which the system is going to work. The prototype is a simple working system that captures the essence of the real system it represents, and it may be refined and redone several times within the iterative process, in order to find the optimum solution for the defined problem.

Prototyping approach needs a software that enables a quick and simple building of a working system. Conventional programming languages and methods, or more adequate software products, like program generators can be used for this purpose. The emphasis is on trying out ideas and providing assumptions about requirement, not on system efficiency or completeness [20]. According to [12], the ideal software prototyping environment has four components:

- a 4GL or other development tool to allow quick creation of the prototype
- well-managed data resources for easy access to corporate data
- a user who has a problem, who has considered the idea of using the new tool, who knows his or her functional area well, and who seeks assistance from data processing
- a prototype builder -- an information systems professional who is versed in using the various development tools and understands the organizations's data resources

The same author also suggests an ideal team size for prototyping -- one user and one builder. Larger teams impose more uncertainty into the problem definition and solving procedure, need more time for coordination, etc... Proper answers to the key output questions represent an important issue of prototyping. These questions are:

- who will receive the output -- what is its planned use
- how much detail is needed -- when and how often is the output needed, and
- by what method

There are three main uses for software prototypes [22]:

- to clarify user requirements. Host users cannot fully describe their current requirements and their future needs. By building, using, and changing a prototype, users can make further decisions about the system they want
- to verify the feasibility of design. A prototype can show the end users of a designed system how the new system would operate, its efficiency and its costs. The end users may find the system inadequate and stop or change it
- to create a final system. Part of the prototype may become part of the production version. Especially, when the system is expected to change very often, it is usefully to use 4GL also for the production version. This makes future use easier

The problems related with prototyping are (i) the need for sophisticated software tools and (ii) that prototyping brings forward only the physical aspects of a system, based on user demands of physical requirements and a quick realization of the needed system. This does not support logical modeling of requirements and solutions. System specifications based on prototypes may inherit inefficiencies and errors of the original system. For this reason, prototyping is most effective when used to enhance, rather than replace, the established analysis process in computer information system development[18].

According to [13] the commercial FOURTH-GENERATION LANGUAGES (4GLs) have provided a significant contribution towards making the concept of prototyping practicable as a methodology for system design and development. They have also helped to create a new information processing environment, referred to as "End-user Computing", despite their primary objective -- to speed up development and maintenance by professional programmers. It is claimed that the productivity in applications development when using 4GLs is 5 to 10 times over that when using third-generation languages, particularly COBOL [17].

The early development of 4GLs started towards the end of the 1960s when time-sharing and DBMS were developed. As the result, three most well-known general-purpose 4GLs software products have become available: RAMIS, FOCUS, and NOMAD. Further, a variety of other user-friendly language interfaces, with elements of non-procedurality, that could be used with the existing DBMS, or previosly developed 4GLs, have been developed (electronic spreadsheets - LOTUS, business

modeling - IFPS, business graphics, statistics, etc.). Another significant event, related to the 4GLs evolution was the development of IBM's DB2, which is a member of a family of relational DBMS products from IBM, all supporting a common relational language called SQL (Structured Query Language). IBM announced its entry into the 4GL market in mid 1986, Cross System Product (CSP).

The features of 4GLs that comprise the functionality that is included in fourth-generation software tools are several [22], like: DBMS, data dictionary (DD), non-procedural language, interactive query facilities, report generator, selection and sorting, screen formatter, word processor or text editor, graphics, data analysis and modeling tools, programming interface, software development library, backup and recovery, links to other DBMS, records and file maintenance, etc. The heart of a 4GL is a DBMS, which can manipulate formatted data records, as well as unformatted text and graphics data. Just as important as the DBMS is the DD, for storing the data definitions used by the 4GL. In contrast to 3GLs, 4GLs employ an English-like syntax, and are eventually non-procedural in nature -- they allow statements to occur in the logical order that a user would think, rather than imposing a sequence required by the computer. Besides an underlying command language, many 4GLs provide a variety of interfaces that help end-users in using them. As far as the results (output) are concerned, some 4GLs generate only single programs (code generators), and produce an intermediate step code in 3GL, usually in COBOL, while others generate complete integrated applications (application generators), and do not produce any 3GL intermediate step code (FOCUS, RAMIS II, NOMAD2,...).

The functions performed by 4GLs vary greatly from product to product. Some 4GL products have highly focused but limited functionality, oriented towards specific applications, like decision-support/modeling tools such as IFPS; graphics generators such as Tell-A-Graf; query and report-generating tools such as DATATRIEVE, and INTELLECT. Some 4GLs are more powerful and comprehensive in terms of their functional capabilities, and represent more nearly integrated software systems rather than "programming languages" as the term is commonly understood [13].

First thoughts, that the emergence of 4GLs means the demise of COBOL have been revised when 4GL-related software (analyzers, generators, and programmer workbenches) began emerged. It is now believed that COBOL will continue to be the dominant language of business into the twenty-first century. The 4GL market has not matured yet. The current effort towards 5GL hardware impose a question whether 4GL software will mature at all or simply blend into 5GL software.

4. THE FUTURE TRENDS IN DSS AND DSS TECHNOLOGY

The short-term trends, from 1986 to 1990, will be an extension of today happenings, with a substantial increase in personal and organizational use of computer technology, as high as 70 to 90 percent increase in computer processing power per year. In less than five years, for instance, there are 8 million users of personal computers with an increase by more than 30 percent annually. But even more important than the above stated advancements are the trends in the change in the application of information technology to the point where users no longer face technical intermediaries between technology and its application. The following major trends can be expected [22]:

- personal computer-based DSS will continue to grow, with spreadsheets and other creativity supporting packages taking more and more functions in analysis and decision making
- growth in distributed DSS, with close linkages between mainframe DSS languages and generators and the PC-based facilities
- group DSS approach, supported by local area networks and group communications services, like electronic mail, will become much more common
- DSS products will incorporate products (tools and techniques) of artificial intelligence, instead of the statistical and management science models of the past. "Intelligent DSS" will assimilate expert systems, knowledge representation, natural language query, voice and pattern recognition, etc..., and will be able to "suggest, learn, and understand" tasks and problems

More user friendliness is expected from the computer technology, such as dialog support hardware (light pens, touch screens), high-resolution graphics, speech recognition and synthesis, menus, windows, etc... It has been proven, for example, that for data manipulation the users strongly preferred voice over keying, because they could continually look at the screen while they dictated operations. It is also believed that expert systems, as part of DSS, will be used more than they are now. Many experts still argue on what is and is not an expert system. For this reason, some authors [22] prefer to speak only about practical "expert-like" systems. These are systems, that capture logic of the application problem by means of a small number or even hundreds of IF...THEN...rules, and can be programmed in any high-level language, such as COBOL, FORTRAN, APL, BASIC, LISP and PROLOG or be expressed in a decision table form. What really matters is that they must help users in making better decisions.

Future trends will show no major changes in basic hardware technologies, though speeds and capacities will be improving steadily at about 10 to 20 percent per year rate, physical rate will be diminishing, while the product life-cycle will remain at about three years. In some areas of applications,for example in transactions processing systems, the trends to replace procedural languages with more powerful tools will be very slow.

In the 1990s the personal computer is going to become a management support facility (MSF). It will be widely used in organizations all around the world because of its technological attributes and low cost. Many decision-makers will have MSF both in the office and at home. MSF will have memory sizes of 10-15 megabytes, and socondary disk storage of up to 250 megabytes. Very user-friendly 4GLs, which integrate computing, modeling, data management, and text processing will be available on all devices. More interesting functions of the technology will be related to the decision-support function: the mainframe databases that are necessary in an organization's transactions processing system will be created and combined with large sub-databases at MSF. Certain systems will be developed to work with their local databases in a problem-finding mode which will help the manager in decision-making. One mode of this type of operation is the use of expert systems in order to locate and solve problems. These problems can be categorized as: dealing with

a crises, evaluating the overall effect of a change, balancing the use of resources, decisions that must be made on resource replacment or acquisition, and trying to forecast the future. Todays largest expert systems involve thousands of logical rules and thousands of objects to which the rules apply. The goals for the 1990s are the expert systems with tens of thousands of inference rules and up to 100 million objects [22].

Technological and organizational changes will also cause changes in the process of management. Three possible manager groups are foreseen:

- information systems managers, who are responsible for the creation, maintenance, and development of the over-all information systems and its resources;
- user managers, who use the centralized information resources, and create, develop and use their personal and functional area information systems;
- senior managers (executive management) who pursue the information systems and resource allocation policies.

The implications drawn from the application of DSS across all forms of enterprise has great importance to higher education. Universities must produce information literate as well as computer literate graduates who can function in the environment of the modern organization. There is, therefore, a need to provide a general course in information systems that would be taken by all students regardless of their major discipline. We need to produce intelligent users of IS.

The implications for universities offering a major in Informatics are even greater. Here is a constant need to modify existing curriculum to reflect the changes in technology from both a hardware and software perspective. In addition new courses will be required to train information majors in the application of technologies. We believe that in many instances this may best be operationalized by a cooperative effort between academics and practitioners. Where information can be shared, ideas refined and students educated all at the same time.

REFERENCES

1. Alonso, J.R.F. SIMPLE: BASIC Programs for Business Applications, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1971.
2. Alter, S. "A Taxonomy of Decision Support Systems", Sloan Management Review, Vol. 19, No. 1, Fall 1977, pp. 39-56.
3. Alter, S. DSS-81 Transactions, Execucom Systems Corp., Austin, Tex. 78766, 1981 and an Interview with Steven Alter, Counsilium Associates, Inc., No. 4, Palo Alto Square, Suite 270, Palo Alto, Calif. 94304.
4. Barle, J., Grad, J., Rupnik, V., and A. Vadnal. Računalniško orientirane matematične metode - VI. IMFM, Ljubljana, Oddelek za matematiko, Raziskovalna naloga, št. 85, 1981.
5. Barle, J., Grad, J., and V. Rupnik. Računalniško orientirane matematične metode - X. IMFM, Ljubljana, Oddelek za matematiko, Raziskovalna naloga, št. 199, 1985.
6. Bonini, C.P. Computer Models for Decision Analysis, Scientific Press, Palo Alto, Calif. 94301, 1980.
7. Dickson, G.W. and J.C. Wetherbe. The Management of Information Systems, McGraw-Hill Book Company, New York,..., Hamburg,..., London..., Toronto, 1985.
8. Gray, P. Student Guide to IFPS, McGraw-Hill Book Company, New York,..., Hamburg,..., London,..., Toronto, 1983.
9. Gremillion, L.L. and A.M. Jenkins. The Effects of Color Enhanced Information Presentations, Division of Research, School of Business, Indiana University, Bloomington, Indiana, USA 47405, Discussion Paper No. 173, 1981.
10. Hackathorn, R.D. and P.G.W. Keen. "Organizational Strategies for Personal Computing in Decision Support Systems," MIS Quarterly, September 1981, pp. 21-27.
11. Jenkins, A.M. MIS Design Variables and Decision Making Performance: A Simulation Experiment, UMI Research Press, Ann Arbor, MI, USA, 1983.
12. Jenkins, A.M. Prototyping: A Methodology for the Design and Development of Application Systems, Division of Research, School of Business, Indiana University, Bloomington, Indiana, USA 47405, Discussion Paper No. 227, 1983.
13. Jenkins, A.M.and B. Bordoloi. The Evaluation and Status of Fourth Generation Languages: A Tutorial, IRMIS Working Paper No. W611, Operations and Systems Management, Graduate School of Business, Indiana University, Bloomington, Indiana, USA 47405, 1986.
14. Keen, P.G.W. and M.S. Scott Morton. Decision Support Systems: An Organizational Perspective, Addison-Wesley Publishing Company, Inc., Reading, Mass, 1978.
15. Keen, P.G.W. "Adaptive Design for DSS," Data Base, Vol. 12, Nos. 1-2, Fall 1980, pp. 15-25.
16. Lucas, H.C., Jr. Implementation: The Key to Successful Information Systems, Columbia University Press, New York, N.Y. 10025, 1981.
17. Martin, J. Fourth Generation Languages: Volume 1, Prentice-Hall, Englewood Cliffs, N.J., 1985.
18. Powers, M.J., Adam, D.R., and H.D. Mills. Computer Information Systems Development: Analysis and Design, South-Western Publishing Company, Cincinnati,..., Palo Alto, 1984.
19. Rupnik, V., Grad, J., and G. Resinovič. Ekonomika informacijskih sistemov, Ekonomska fakulteta Borisa Kidriča, FCI, Raziskovalna naloga po pogodbi z Iskro, ZORIN TOZD CAOP, Ljubljana, 1985.
20. Senn, J.A. Analysis and Design of Information Systems, McGraw-Hill Book Company, New York,..., Hamburg,..., London,..., Toronto, 1984.
21. Sprague, R.H., Jr. and E.D. Carlson. Building Effective Decision Support Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1982.
22. Sprague, R.H., Jr. and B.C. McNurlin. Information Systems Management in Practice, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1986.

ARTIFICIAL INTELLIGENCE EXPERIENCES ITS OWN BLINDNESS

INFORMATICA 3/87

UDK 681.3:519.7

Anton P. Železnikar
Iskra Delta, Ljubljana

Abstract. This essay deals with the following topics: today's controversies of artificial intelligence (stupidity, blindness, traditional hardness, informational statics, breakdowns, non-theoretical approaches, bewilderment, overselling, culture of the hacker, non-intelligence), information and its background and foreground, the nature of blindness and breakdown, some stupid manners of ai, crisis of ai, structuring and organizing information, artificial and natural intelligence, intelligent machine, intelligent program, and the new perspective of ai. This essay stresses the importance of information, which has to be understood as a spontaneuosly arising, developing phenomenon in which the phenomenon of intelligence can be embedded.

> ... But when we consider the essence of technology, then we experience Enframing as a destining of revealing. In this way we are already sojourning within the open space of destining, a destining that in no way confines us to a stultified compulsion to push on blindly with technology or, what comes to the same thing, to rebel helplessly against it and curse it as the work of the devil. Quite to the contrary, when we once open ourselves expressly to the essence of technology, we find ourselves unexpectedly taken into a freeing claim. ...
>
> M. Heidegger: The Question Concerning Technology, pp. 25-26.

0. Introduction

Artificial intelligence (ai for short) is the study of complex information-processing problems that often have their roots in some aspect of biological or neurophysiological information processing. The goal of ai is to identify and solve such information-processing problems. However, today's ai is essentially enframed in a western rationalistic tradition and through this, it is reaching the stage of its blocking blindness which has to be broken down for the sake of its own advance. This essay shows one of the new perspectives which is based on the informational footing of intelligence.

In regard to ai, its state of the art and its applicability, the following controversial statements can be read and heard: Artificial intelligence meets natural stupidity. The tendency has been to look for ai solutions for their own sake. So far, ai has not had much of an impact on our lives. Artificial intelligence needs to be established as a theoretical discipline, such as physics or biology, so that large, usable and reliable expert systems can be built. Today's expert systems are static, the man-machine interface is crude and there is little integration with the rest of the software world: specification, testing, reliability, and documentation are all black arts. Artificial intelligence has singularly failed to do anything useful for anyone involved in the wealth-creating economy. Artificial intelligence has been grossly oversold, leaving new users in a bewildered state. Fears that ai will interfere with current working practices and replace human experts are based on the misconception of how advanced ai really is: systems are nowhere near their potential. Market forecasts are wildly optimistic: artificial intelligence is still mostly technology-driven rather than market-driven. Artificial intelligence technology will not be taken up widely until users are prepared to state what they are doing with it, and what it can do for productivity. And although this may sound naive: Expert systems could speed the provision of true equality and democracy.

On the other hand, researchers (e. g., Raymond Reiter) still believe that ai can be put on a formal grounding, and that there are general principles of intelligence to be discovered, even if human thought does not in practice operate on those principles. One reason to have theories, general theories, is that they can be passed on to the next generation, who will then modify the theory, test it out, try to find holes in it, and explore its consequences. If there is no science of artificial intelligence, if it is strictly an encyclopedia of highly special-purpose representations, if there is no commonality across different modalities of intelligence, than the best that can be done is to build very brittle and fragile programs, the principles of which are not translatable from one generation to the next. Even if it is not possible to prove, some researchers in ai and cognitive science believe that human intelligence does in fact work on general principles. Such general principles of

intelligence could be the informational embedding and the informational arising. In this context, the question of artificial intelligence's blindness and its possible breakdowns can be put onto the horizon of the discussion which follows.

1. Information, Its Background, and Its Foreground

It is believed that higher cortical functions and forms or modes of intelligence (the general and factorial ones) are grounded as fundamental neurophysiological forms and processes in a specific biological substance. On the level of a biological substance and processes within this substance, different sciences are introducing a preferably neutral grasping of understanding these processes, using the notion which they call information. Through the centuries, in the development of philosophy, the notion of information has been the understanding of different forms and processes. In brain research and neural science, neurophysiological substance and neurophysiological processes are concerned with a particular phenomenology, which can be understood as neuroinformatics.

Informatics is the neural, scientific, technological, and social discipline, the central and main subject of which is information, informational forms, and informational processes. In the abyss of a phenomenon understanding, the ground of this abyss is information. Information is recursively embedded in existing and arising information. However, on the level of a living being, a being's information is falling into its own abyss, whose bottom is neurophysiological. It seems that information of a being is embedded in a being's neurophysiological system. At the current stage of scientific development it is not possible to construct a platform connecting the top and the bottom of a being's informational abyss, to determine the embedding of information in a structured and organized neurophysiological substance. However, this does not mean that the neurophysiological bottom, which represents the surrounding for embedded information, cannot be brought into informational enlightenment. At this point, information is circularly concerning information. Therefore, grasping of information, in which intelligence is embedded, has to be understood by information itself. In this basic arising of the problem of information, it becomes necessary to explicate and to articulate the question of information in a more definite and constructive way.

In this essay three informational levels can be described: the neurophysiological, the informational, and the intelligent one. Even though the background of information is neurophysiological, it can be illuminated informationally. Information is circular concerning information. The foreground of information is intelligent because informational forms and informational processes are informational constructs and informational compositions explicating various informational properties. It can be understood that intelligence is embedded in information and that information is the sought environment of intelligence. Thus, investigation of informational phenomena is becoming fundamental and relevant for the design of future informational subjects, informational objects, or information machines, all of which possessing some intelligent properties.

2. The Nature of Blindness and Breakdown

How does man understand cognition and how is this cognition blinded? On the way to human cortices, sensory information is extensively transformed by neurophysiological conversion, internal world model's and emotional filtering and modulation, and not lastly, by individual processes of reflection and abstraction. The basis for this type of understanding is being-in-the-world (Heidegger). Man's ability to treat his own experience as involving present-at-hand objects and properties is derived from a pre-conscious experience of them as ready-to-hand. When man is being thrown in a (new) situation of acting with his pre-reflective experience, he does not have the opportunity or need to disengage and function as detached observer. In this situation, reflection and abstraction are not the basis of a man's everyday action. Whenever a man treats a situation as present-at-hand, analyzing it in terms of objects and their properties, he thereby creates a blindness. His view is limited to what can be expressed in the terms he has adopted (Winograd, Flores). Reflective thought is impossible without the kind of abstraction that produces various phenomena (informational processes) of blindness.

Information of objects and properties is not inherent in the world, but arises on the cortical level only in events of breaking down in which it becomes present-at-hand (Heidegger). A hammer as such, being used by someone engaged in driving a nail, does not exist. It is a part of the background of readiness-to-hand that is taken for given without explicit recognition as an object. The hammer presents itself as a hammer only when there is some kind of breaking down or readiness-to-hand (for example, if it breaks or cannot be found). In a more general way, a state of pre-reflective blindness can be changed only through events of breaking down.

The nature of blindness in the field of artificial intelligence is resting on the traditional (pre-reflective) givenness (experience) of rationalistic thinking, which is grounded in mathematical methodology, algorthmic approach, formal languages, deductive and inductive inference, and other hard sciences, all of which are characteristic ways of thinking which use the left-hemisphere of the brain (or isolated to much from the right hemisphere). The blindness of a discipline can be broken-down through new philosophical orientations whose groundings are essentially different from those of rationalistic tradition. Artificial intelligence as an encyclopaedic and rationalistically hard-oriented discipline is not yet understanding and recognizing the real and soft problems of natural intelligence, so it is not in the position to set new intelligent methodologies or construct programs or machines which could be, in fact, intelligent. Intelligence within artificial intelligence is not treated informatically at all. It is not being considered that intelligence is a particular or composed form or process of information and therefore the question of information, of its Being, within artificial intelligence has never been exhaustively raised and questioned.

The question of information is generating a questioning, and through this questioning it is delivering the interrogated, which is illuminating the notion of information, to obtain sight into the essence of information. This questioning represents one of the possible breakdowns in comprehending of intelligence.

Hence, intelligence is coming on the way to information; it is coming into existence as information, as a particular informational form or informational process following the informational principles of structuring and organizing, i.e. of embedding and arising.

3. Some Stupid Manners of Artificial Intelligence

Some manners of ai showing dullness of mind are described in D. McDermott's 'Artificial Intelligence Meets Natural Stupidity'. As a field, ai has always been on the border of respectability or of crack-pottery. It has explored weird ideas, in the hope that pursuing them was the only way to make progress. The necessity for speculation has been combined with the culture of the hacker in computer science to cripple the ai self-discipline. The tolerance of sloppy thinking led ai to repeat mistakes over and over.

A relevant source of simple-mindedness (blindness without occurring breakdowns) in ai programs is the use of wishful mnemonics like 'understand', 'recognizer', 'understander', 'goal', 'resolver', 'paramodulator', 'context', 'natural-language-interface', 'is-a', etc. to refer to programs and data structures. To a great degree, such programs are unsolved problems rather than solutions. Although a researcher does not have the means to understand an 'understander', he thinks he can come closer to a solution if he innocently states the question of 'understand'. So, he is misleading himself and others that through time this identifier will lead to the wishful solution. Many examples of wishful mnemonics come to the mind. For example, GPS (short for General Problem Solver) caused a lot of needless excitement and distraction and its proper identifier should have been LFGNS (Local-Feature-Guided Network Searcher).

A pervasive sloppiness in ai thinking is the tendency to see a natural source of problems and solutions in natural language. The obsession with natural language seems to have caused the feeling that the human use of language is a way to the cognitive psyche. Natural language is only a particular form of information, only a part in a being's entire information, disturbing a being which is a speaker or a listener in an informationally subjective way. In no way is a natural language close to the language of a being's thought, to a being's complex internal informing. Language nuclei in the cortices are only particular areas in the entire brain structure, only straightly specialized co-processors in the parallel brain configuration. Natural language is a kind of secondary language, a mediator between a being's internal information and a being's environment. As of now, there is no idea at all how being experiences its thoughts in images and words, how it experiences its thoughts at all.

4. What is the Crisis of Artificial Intelligence

Crisis is a controversial information occurring between the existing blindness and the arising breakdowns which have to interrupt the blindness' ruling and which have to start a new orientation. Crisis is an informational process of deciding how to progress into blindness' context and how to change this context accordingly to breakdowns' requirements.

Ai's blindness is rooted in the western rationalistic tradition which is appropriate for ideally oriented and arbitrarily simplified and formalized sciences like mathematics, however it is not appropriate and sufficient for sciences concerning living information systems or even intelligence on the level of the human cortex. Blindness of ai's existing orientation lies in the belief that today's algorithmic approaches which are evidently non-intelligent will, in time, reach the level of intelligence by gathering more and more data and more and more algorithms, altering them in an intelligent way. It is not clearly understood yet that artificial intelligence qua intelligence needs an arising machine and program substance. This last part of the previous statement represents the framework of the most relevant breakdown and requires that the foundations of today's ai have to be changed in a way to become informationally founded.

5. Structuring and Organizing Information

Information informs. Informing, which is an action of producing and receiving information, is governed by two basic informational principles: informational embedding and informational arising. Arising of information generates new information which is denoted as counter-information. Thereupon, this new information becomes a part of informational embedding ready to be used for the generation of subsequent counter-information. What are the forms and processes for structuring and organizing information? What is information which structures and organizes information?

Information is circularly concerning the structuring and organizing of information. These informational self-structuring and informational self-organizing principles have to be explained for comprehension of Being of Information. There are two basic informational principles which constitute structuring and organizing of information qua information: embedding and arising. From everyday experience it is known that a new information must be embedded in the existing one, otherwise its informational comprehension would be not possible. Thus, a new information is informationally 'expressed' by the existing one. However, new information informs in a new way and after its arising it is becoming a part of a new informational embedding. In this way, it is possible to enlarge and to enrich the existing informational embedding by informational arising. Embedding and arising of information are both simultaneously and dynamically interactive informational phenomena, which change, enlarge, or semantically enrich the realm of informational embedding. Evidently, through informational arising, informational embedding is also arising changing, enlarging, or enriching the realm of informational.

Informational embedding and informational arising can be explained statically by the example of a computer system. A computer system is applicable, user-friendly, or ready-to-hand if its usage has reached an adequately symbolical or linguistically appropriate level of application. This readiness-to-hand is a consequence of an alternate application of system embeddings and system arising (in this case, adding computer hardware and computer software) by an outward (system engineering) action of arising. In this way, a computer system is readied-to-hand by readying-to-hand when a new functional module is embedded into the existing computer structure and through this embedding it is becoming a part of a new structure, changing, enlarging, or enriching

the previous structure and enabling further additions of functional modules.

6. Artificial and Natural Intelligence

Intelligence is a conjoined principle resulting from a specifically informational, spontaneous, and intelligently oriented structuring, organizing, and arising of information. Natural intelligence is always an arising principle which is embedded each time in the existing intelligence, where intelligence itself is embedded in a being's entire or total information. Intelligence is coming into existence through several informational arisings and informational embeddings, where the arising structure and the arising organization are determining the course or orientation of intelligence development up to the given time slice. Thus, intelligence is nothing more than presently structured and organized information in the informationally oriented interaction of arising and embedding. The course or orientation of intelligence generation is by itself information which is influencing the coming of intelligence into existence. Accordingly, the principle of intelligence can be stated in the following way: arising of intelligence, or coming of intelligence into existence, is the consequence of informationally and interactively oriented structuring, organizing, and arising of information including intelligence.

On the contrary, artificial intelligence is not based yet on the informational principle of arising in a dynamic, informational way. Similarly, the embedding of ai structure and ai organization is not arising in a dynamic manner. AI is not even arising in a way similar to the lowest informational forms and processes of living information. The consequence of these static ai principles of arising and embedding is that ai is intelligent only through its outward maintenance, however not by itself.

7. Intelligent Machine and Intelligent Information

The objectives of ai in the future are to enable the construction of intelligent machines for which intelligent information can be applied. Since intelligence is the phenomenon which is embedded in information, an intelligent machine is nothing more than informational machine, and intelligent information is a consequence of informational programming. An intelligent machine is embedded in an informational machine which comprises an informationally dynamic architecture (being switched by signals, messages, or information). An intelligent program must have the property of informational arising during its execution. In this respect, the intelligent program is varying, arising, or vanishing during its execution, so, it can not be developed by means of today's programming methodology (e. g., structured programming). Both, intelligent machines and intelligent programs have an informationally dynamic structure, organization, embedding, arising, counter-Informing, etc.

8. The New Perspective of Artificial Intelligence

This new perspective of ai is in no way pessimistic, although it is conditioned by an entirely new, informational foundation. This foundation is the most important breakdown essentially influencing the progress in a new orientation. It is becoming evident that short term goals of approaching intelligent computer systems are not reachable. It is becoming even evident that the splitting of today's computer systems into non-intelligent and intelligent ones will become economically reasonable. Intelligent machines and intelligent programs have to be developed in their own, informational way by technological reconstruction and intelligent specialization of today's computer system. In this respect, the aim of ai today is not so much in searching for intelligently impossible solutions on existing computer technology as searching of new ways, on which new, dynamic technological architectures and new, intelligent tools for information programming have to be discovered.

References

M. Heigegger: Being and Time. Harper & Row. New York, 1962.

M. Heidegger: The Question Concerning Technology and Other Essays. Harper & Row. New York, 1977.

D. McDermott: Artificial Intelligence Meets Natural Stupidity. Mind Design: Philosophy, Psychology, Artificial Intelligence. (J. Haugeland, Editor). The MIT Press, Cambridge, Mass, 1985.

T. Winograd, F. Flores: Understanding Computers and Cognition: A New Foundation for Design. Ablex Publ Corp, Norwood, NJ.

A. P. Zeleznikar: On the Way to Information. Informatica 11 (1987), No. 1, 4-18.

A. P. Zeleznikar: Information Determinations I. Informatica 11 (1987), No. 2, 3-17.

A. P. Zeleznikar: Raziskave računalnikov in informacije v naslednjem desetletju. Informatica 11 (1987), No. 2, 57-59.

* * *

Solutions are Being Sought for Their Own Sake. Computer Weekly. Oct 9, 1986, p. 30.

Putting Intelligence on a More Formal Footing. Computing. Oct 16, 1986, pp. 28-29.

A PRODUCTION PROBLEM INTERACTIVE PROTOTYPE OF LINEAR PROGRAMME

INFORMATICA 3/87

UDK 519.852

Janez Barle, Janez Grad, Džordž Krstič*
Ekonomska fakulteta Borisa Kidriča, Ljubljana
***Iskra Zorin, TOZD CAOP, Ljubljana, Yugoslavia**

In the paper we describe our approach in developing a new program package comprising different models of linear programming which all together build up a sophisticated system for production planning and decision making. By means of it the necessary information will be generated which the top management of Iskra enterprise, Ljubljana (one of the manufactures of electronic and computer equipment in Yugoslavia) is asking for. The input data are mainly retrieved from a data base and partly generated by means of a matrix generator. By generating different application matrices with different objective functions various production plans can be studied. The specific characteristics of the discussed production problem are (i) the need for balancing exports and imports and (ii) the ability to cope with a very high rate of inflation and very high rate of bank interest charges.

+Presented at "Deutsche Gesellschaft für Operations Research - Tagung 1986", Ulm, W. Germany, 24. - 26. Sept. 1986.

1. INTRODUCTION

Optimization methods of manufactured assortments are generally not used frequently enough within the basic production process activities in most industrial establishments. Defined in a common linear programme form these methods could be particularly useful in those industries where they offer the market many different products or their variants and where the main proportion of the products, aimed for an unknown purchaser, is to be stored in a warehouse for a certain period of time (KRSTIČ, 3). There are two main reasons why the usage of the production problem linear programme has not been adequate within the establishments where it could have proved both possible and profitable. One reason is the lack of knowledge of optimization methods among the management who therefore are not in a state to draw and build up the necessary schemes and models of possible applications. The second reason is the rather awkward and demanding presentation of the LP input data and the complex interpretation of the LP output results which makes them hard to understand and explain to the end-users. Three years ago in the Iskra enterprise, Ljubljana, we drew up adequate models and organized the necessary education courses for end-users. We have been running on the computer some optimization methods for single members (factories) of the enterprise since then. Unfortunately all the computer programs are batch orientid and are not linked together with other parts of the information system, despite the use of some interactive equipment. This leads to a substantial redundancy in data within the Iskra information system.

A new programme package is being developed at present comprising different models of LP which all together build up a much more sophisticated system for production planning and decision making. By means of it the necessary fundamental information will be generated which the top management of a manufaturing company is asking for. The input data are mostly extracted from a data base and partly generated interactively by means of an application orientated matrix generator. By generating different aplication matrices with different objective functions, various production plans could be studied. The optimal solutions obtained by LP are further processed by a computer aided expert system. In the man-computer interface the most suitable production assortment is selected. Useful sugestions for the elimination of production bottle-necks and other complementary information could also be obtained. The management and planning functions that link together the financial and production parts are two important domains of the program system application.

The present LP program package, which is written in the APL programming language is of a prototype form. The final version will be in PASCAL. It solves linear programmes with up to 200 constraints and with up to 500 variables. The revised simplex method is applied where the basis matrix is presented in product form with spike selections. Great attention has been paid in order to make the package as user friendly as possible and less attention has been paid to the program's efficiency in execution.

In this paper we describe our approach and experience in developing the necessary software for solving LP in this particular field of application.

2. CHARACTERISTICS OF THE PRODUCTION PROBLEM MODEL

Let us first describe the general production (assortment) problem of linear programming.

Suppose that a company produces n products P_1, P_2, ..., P_n by means of m different elements of the available production resources, such as machines, human capacities, financial means and other, the amounts of which are restricted by b_1, b_2, ..., b_m within the discussed period of time. The consumption of each element of resources per unit of the produced product P_k is known and we denote it by a_{jk}, for all j = 1, 2, ..., m and k = 1, 2, ..., n. There exist upper and lower production quantity bounds for each product due to the limited production resources, marketing restrictions (possibilities) and the signed contracts that impose certain obligations in production planning. The problem that arises most often is to define such an assortment and the corresponding quantities of the products which assure the maximum value of the profit after sale. Sometimes we try to optimize the net profit c_k of product P_k, i.e. difference between its net selling price and its direct production costs.

Denoting by L_k the lower, by U_k the upper possible and by x_k the optimum production quantity of P_k we obtain

$$L_k \leq x_k \leq U_k \text{ , for } k = 1, 2, \ldots, n$$

or in a matrix notation

$$L \leq x \leq U$$

Similarly we may introduce

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{bmatrix}, \quad A = \begin{bmatrix} a_{11} \cdots a_{1n} \\ a_{21} \cdots a_{2n} \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ a_{m1} \quad\quad a_{mn} \end{bmatrix}, \quad c = [\, c_1, c_2, \ldots, c_n]$$

and define the general production problem LP in the following way:

Find (compute) the solution of production quantities vector x, satisfying relations (1) and

$$Ax \leq b$$

for which the value of the objective function f(x) is maximum, where

$$f(x) = (c,x).$$

We can solve this problem by means of the general simplex method.

The specific characteristics of the production problem LP for the needs od the Iskra enterprise take into account the following two characteristics of the present state of the economy in Yugoslavia:

(i) The need for the closest possible balance between exports and imports within each particular enterpise
(ii) Very high rate of inflation and very high rates of bank interest charges (40 % - 70 %).

The possibilities of coping with the conditions that accompany the two characteristics differ very much among different enterprises. We explain here only briefly the basic characteristics of the foreign currency balancing assortment problem which incorporates both the resources and users of foreign currency at the same time. More details on the subject are given in (KRSTIČ and KLUČAR), 4).

The basic inequality which we add to the system of constraint functions is

$$\sum_{k=1}^{n} x_k \, (E\text{-}I)_k \geq D$$

where $(E\text{-}I)_k$ stands for the export - import trade balance of product P_k exported to some foreign customer. The same product may differ in the export price E and the amount (price) of the imported raw material I from one country to another one. We therefore consider it in the LP as a different product for each different country. The values of the export - - import trade balance for all the products are stated in the same foreign currency unit, usually in DM or in US dollars. Product P_i produced for the Yugoslav market would have a negative export - import trade balance ($-I_i$), according to the above definition. D in the above relation means the necessary surplus in foreign trading which the enterprise needs for some other purposes such as the import of the necessary equipment, advertising in foreign countries and similar.

The example of treating each product as a different one for each diferent country shows that we have to deal with a large amount of LP input data (usually we optimize the production programme twice a year, accomplishing 3-4 consecutive computer runs each time and enabling the users to exchange some of the parameters). Here we use electronic spreadsheets as an additional and helpful resource for presenting the original data in the LP input form. Nevertheless we are nowadays not satisfied with this kind of approach. We have decided therefore to develop a new program package which is better orientated to the end-user. It is based on the results and experiences of the successfully performed production planning model adjusted to the real environment and giving reasonable financial effects both in Yugoslav and foreign currencies.

3. CHARACTERISTICS OF THE PRODUCTION PROBLEM INTERACTIVE-TYPE MODEL (PACKAGE) DESIGN

The discussed model (program package) consists of three modules:

- LP input data preparation module;
- LP module and
- postoptimization module.

In the following we describe the basic characteristics of each of these modules.

3.1. LINEAR PROGRAMME INPUT DATA PREPARATION MODULE

It consists of three parts that handle the following activities:

- retrieval of data from the Iskra enterprise MIS data base;
- interactive input and modification of the additional data;
- building up the LP input data matrix.

In order to retrieve data from the data base the user states the name of the product and the corresponding attributes that he wants. New products and new attributes can be added into the data base and modifications to the retrieved data (prices to-come, expenses, ...) can be performed interactively. The user can further state and define his conditions and demands with regard to restrictions on the production problem solution. He either accepts the already existing restrictions within the LP or imposes some new ones.

From the user standpoint there exist two ways (modes) of LP data preparation:

(i) The general mode which enables the common approach to the LP process definition (min of max value of the objective function, optional relation operators of the constraint functions, etc.).

(ii) A mode oriented solely towards the production problem LP. It is user-friendly approach with additional references (options) explaining the management view and demands on standards and the policy which should be taken into account within the production process optimizing procedures. The computer/user dialog includes questions and answers about the chosen objective function which should be added to the LP data matrix of the constraint functions. In a similar way the user also gets the list of existing and available production resources which he may use in the LP programme. Here too he may insert different relational operators.

3.2. LINEAR PROGRAMME MODULE

The most efficient methods for solving LP problems are different variants of the simplex algorithm. Due to the dimensions of our problem we could not use the standard simplex method. It was necessary to employ the revised simplex method with basis matrix in a product form (MURTAGH, 5). However, the expected dimensions of our problem don't require the use of the most sophisticated variants of this method. The most advanced feature used in our program is probably the algorithm for periodic refactorisation of the basis matrix with the "bumps and spikes selecting procedure".

We must mention, however, that this module programmed in APL shows relatively high inefficiency in execution compared with a similar FORTRAN program which is about ten times faster.

3.3. POSTOPTIMISATION MODULE

Similarly to the LP input data preparation module, the postoptimisation module enables two modes of operation. In the general mode, the common postoptimality (sensitivity) analysis is performed. It includes:

a) computing the intervals within which the values of particular coefficients of the objective function can range and yet maintaining the same optimal solution (providing all other elements remain the same),

b) computing the intervals within which the values of particular coeffiecients of the right-side vector b can range and yet maintaining the same optimal basis (with changed values of the basic coefficients of x).

Other standard types of LP output (MURTAGH, 5) are also provided.

For our particular application another mode of operation is available which is more user-friendly. It expresses all LP output data in terms of production and finance. Using this mode production bottle-necks because of technological constraints of foreign currency availability can be studied much more easily by the average end-user. Such interpretations also raise the level of user interest and expertise in LP models within Iskra.

4. PROTOTYPING APPROACH

We extensively used the prototyping approach when working on these problems. Prototyping (BOAR, 1) is a relatively new method for extracting, presenting and refining a user's needs by building a working, user-friendly model of the ultimate system quickly and in context. The prototyping approach, by incrementaly refining the model, leads to a better understanding of the problem and hence helps in the development of effective and efficient solutions.

There are four basic stages in the prototyping life cycle:

(i) Identify user's needs - define the problems and define the functions and data needed to solve them.

(ii) Develop an interactive model - quickly create a preliminary working model that incorporates the key items identified in the previous step.

(iii) Demonstrate the interactive model - while demonstrating the small system encourage the active role of the user - to add to and improve the system.

(iv) Revise and evaluate the system - the prototype is alternately revised and implemented in the user's environment until the system is acceptable to both users and developers.

For prototype building we use the APL programming language. This is a high level language which is particularly useful for the quick development of interactive programmes (GILMAN, ROSE, 2). Its ability to perform vector and matrix operations make it also a powerful tool for testing mathematical algorithms. The main obstacle for wide use of APL is its inefficiency in execution. Nevertheless APL proved to be very useful as a prototyping language for our particular type of application. We feel that our APL based prototype is a very good basic bor the development of a production solution, which will be in the PASCAL language.

5. CONCLUSION

Our project to build an LP production problem interactive prototype is not finished. Up to now our main attention was devoted to building an adaptively designed optimization model and appropriate LP module. The data preparation module and postoptimisation module are not yet fully integrated into the system.

The experience obtained in solving production problems for both particular factories and the whole enterprise has encouraged us to continue and expand this project.

REFERENCE

1. Boar B.H., "Application Prototyping - a Requirements Definition Strategy for the 80s", John Wiley & Sons, New York, 1984.

2. Gilman L., Rose A.J., "APL: an Interactive Approach", John Wiley & Sons, New York, 1976.

3. Krstič D., "The Use of Univac LP and MCS Packages", in: UUA/E Conference Technical Papers, Wien, Nov. 1987: The Effect of Communications Policy and Technology on Computer Planing, UUA/E, London, 1979, p. 346.

4. Krstič D., Klučar N., "Optimiranje proizvodnih programov ob upoštevanju pokritja deviznih potreb z izvozom", Organizacija in kadri, 18 (1985), 3-4, p. 218 (in Slovenian).

5. Murtagh B., "Advanced Linear Programming: Computation and Practice", McGraw-Hill, New York, 1981.

UDK 519.68

John D. Freyder
Iskra Delta, Ljubljana

Reasoning Simulation Programs offer no intelligence, but rather a high form of abstraction from machine level which allows clearer and more concise algorithms and data representation for a new class of problems. In general, rule based systems which model a logical deductive process may be divided into two categories; forward chaining and backward chaining systems. Two major differences between these two approaches are the type of information (i.e. facts) generated and the flexibility of search strategies used. Methods to increase efficiency and to control the path of inferencing are discussed. These methods are largely modeled after those used in automatic mathematical theorem provers and may include subsumption, directive reasoning, search control, variable free resolution and proof by contradiction.

Introduction

Programming languages and systems*, which are based on Predicate Logic, offer a wide number of advantages over "conventional" sequential programming languages. This is largely due to their high level of abstraction from the workings of a bare machine. One, who is familiar with both Assembler and Pascal, knows the relative advantages Pascal has in problem defining, algorithm development, and logical construction for flow of control. These advantages come from the structure of the language itself (its form of abstraction from machine code) which defines how the language represents and manipulates data. The form of abstraction for programming languages or systems, which are based on some form of logical reasoning, is not only one step higher than conventional programming languages, but also one step away from sequential programming languages. That is, these languages are declarative rather than sequential. Also, they generally incorporate (implicitly or explicitly) a high degree of recursion.

Systems, which simulate various forms of human reasoning and logic, have opened and are employed in a number of new areas. Since the late 60's, development in programs which where based upon Mathematical Logic and where designed as automatic theorem provers have helped lay one foundation for reasoning simulation programs. Programs which use this foundation employ some form of Predicate Calculus and may be viewed as rule based systems. Although languages which incorporate inferencing mechanisms and strategies are currently available (Prolog, OPS5, ect.), understanding the structure and logical formalism which such a language incorporates is very valuable. For systems based on Predicate Logic, various notations are available, e.g. polish notation, frames and semantic nets [Bun83]. However, the clearest and most concise notation for a short discussion is predicate (or functional) form.

* the term system is used because, although programming languages based on Predicate Logic (such as Prolog) do exist, there are other options, such as ITP (interactive theorem prover developed at Argonne National Laboratory), which are far more extensive then a programming language.

General Background

In functional form, there is the predicate and its corresponding arguments (the predicates t and f which represent true and false, respectively, are also included). A predicate represents a relation between its arguments and returns a truth value. The general form for a predicate is P(X1,X2,...,Xn) where n => 1 and for i = 1,2,...,n each Xi is either a constant, a variable or a function which returns a value. The convention, in which variables are represented in upper-case and constants are in lower-case, is used. Because all variable free predicates within the system (i.e. predicates which only have constants as arguements) are always assumed to have the truth value t, they are often referred to as facts. Note that the system raises no objections to false facts. The following four examples represent predicates.

```
1: =(4 4)
2: connect(wire1,wire2,wire3)
3: node3(edge4,Y)
4: connect(X,Y,Z)
```

From a set of predicates, rules are constructed in order to deduce new relations between objects (i.e. new facts). A rule consists of an antecedent and a consequence and is built by using the three boolean connectives v, &, - (or, and, not, respectively) and the implication arrow, --->. A single predicate or the negation of a single predicate which is in a rule (or in a clause) is often referred to as a literal. Deductions are made by choosing a rule and attempting to satisfy each literal in the antecedent of the rule. If the antecedent can be completely satisfied then the consequence is generated and the newly deduced facts are added to the current set of facts. The satisfying of a literal in the antecedent is achieved by instantiating the unbound variables in the literal with the values of the arguments of a valid matching predicate in the current set of facts. The general form of a rule is P----> Q, where both P and Q consist of one or more literals. As an example, given the following two rules:

```
1: P1(X,Y) & P2(Z) & P3(X,Z) --->
   R1(Y,Z) & R2(X)

2: R1(X,Y)  ---> Q1(X,Y)
```

and the following facts,

```
3: P1(a,b)
4: P2(d)
5: P3(a,d)
```

The two facts which are deduced with Rule 1 are listed below. The instantiation of the variables found in the rule is listed to the right.

```
6: R1(b,d)      (X a, Y b, Z d)
7: R2(a)
```

and then from Rule 2 the following fact is deduced

```
8: Q1(b,d)     (X b, Y d)
```

Note that in order to make the deduction using the second rule, the deduction using Rule 1 (#6) was needed first.

Given a set of facts and a set of rules, various paths to a solution are often possible depending on the order in which deductions are made. The order in which the rules are used generally determines the path of inference taken. That is, the order in which the antecedent of the rules are satisfied directly determines the order in which the new facts are deduced. This order is usually determined by choosing a predicate in some manner with which the antecedent of matching rules may be satisfied. If one views the path of inference as a tree, then the path taken in order to reach a solution represents a search of the tree. In this manner, depth first and breadth first searches, among others, may be implemented in finding a solution. Forward chaining and backward chaining are the two methods generally implemented for choosing a rule from the set of rules, and thus, in part, influence the path of inference. With forward chaining, rules are choosen by matching a literal in the antecedent of a rule. For backward chaining, rules are choosen by matching a literal in the consequence of a rule. As one would expect the choice between which method to implement implies a trade-off between flexibility and efficiency.

Backward Chaining

The goal directed approach, implemented by a depth first search, often allows backward chaining great efficiency. The path of inference begins with a "goal" (i.e. the deduction which is desired and which represents the root of the tree). Given the goal, the consequence of each rule is matched against the goal. If a rule contains a literal in its consequence which matches the goal, then an attempt is made to satisfy each literal in the antecedent of this rule. In this attempt to satisfy the antecedent, each literal in the antecedent is sequentially taken as the current "goal", all necessary variables are instantiated and an attempt is made to satisfy it. If all literals in the antecedent are satisfied then the consequence is generated. Due to the fact that the rules are selected by their consequences, often the robustness of reasoning is cleanly insured when rules are written in a form such that the length of each consequence is not longer than one (e.g Horn clauses as used in Prolog).

Consider the following example of a depth first search strategy implemented by backward chaining. Given the following goal, facts and rules:

```
Goal:  Goal(X,Y)

Facts:
1) P1(a)
2) P1(b)
3) P1(c)
4) P4(d)
5) P4(e)

Rules:
6) P1(X) & P2(Y) ---> Goal(X,Y)
7) P3(X) ---> P2(X)
8) P4(X) ---> P3(X)
```

The tree in Figure 1 represents a depth first strategy to generate the six possible deductions which satisfy the goal and are listed below. The solid edges represent that part of the tree which has been searched, whereas the broken edges represent the nodes which will be searched.

```
9)  Goal(a,d)    11) Goal(b,d)    13) Goal(c,d)
10) Goal(a,e)    12) Goal(b,e)    14) Goal(c,e)
```

Forward Chaining

Although efficient, backward chaining only generates facts (i.e. information) which satisfy a specific goal. Other information which does not lie directly on the path of satisfying the goal is never generated; although it may be very informative and pertinent. Such information may be useful in problem solving where goals are either not clearly defined or very general. With forward chaining, the facts generated may tend to be more general. This is due in part to rules which may be represented differently and which yield more general facts from their consequences than is possible using backward chaining. Also, due to the fact that a forward chaining system lacks a specific goal to satisfy, it deduces all possible deductions given a set of facts and set of rules.

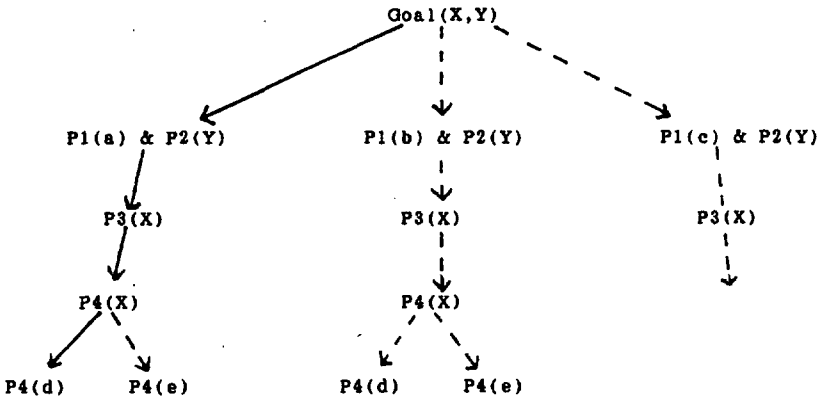


Figure 1: Depth first search with backward chaining.

Rules and Facts

Forward chaining allows greater flexibility in search strategies and generates more general information at the cost of greater inefficiency. With the absence of the goal-directed approach, as described above in backward chaining, Horn clauses (and other forms where the consequence is a single literal) no longer offer an advantage as a form for the rules to be written in. Rather, removing the restrictions Horn clauses impose on clausal form is more suitable and advantageous when using forward chaining. A clause is simply defined as a disjunction of literals and the length of the clause is equal to the number of its literals. If clausal form is loosely implemented, then for each rule (as represented in the common P ---> Q form) the antecedent consists of a conjunction of literals, whereas the consequence may be either a disjunction of literals or a conjunction of literals. The key to translating these rules into clausal form is that P ---> Q is logically equivalent to -P v Q. Thus, each rule of the form P1 &...& Pn ---> Q1 is logically equivalent to the clausal form -P1 v...v -Pn v Q1. Extending this, each rule of the form P1 &...& Pn ---> Q1 v...v Qn is logically equivalent to -P1 v...v -Pn v Q1 v...v Qn (this is known as Kowalski form). Finally, each rule of the form P1 &...& Pn ---> Q1 &...& Qn may be viewed as a compact notation for the logically equivalent set of rules (which are Horn clauses)

```
-P1 v...v -Pn v Q1
 .         .   .
 .         .   .
-P1 v...v -Pn v Qn.
```

Note that literals in the antecedent may also be negations of predicates. Hence, P1 & -P2 & P3 ---> Q1 v Q2 is often translated into clausal form as -P1 v P2 V -P3 v Q1 v Q2 which, by association, is equal to -P1 v -P3 v P2 v Q1 v Q2. This represents the logically equivalent rule P1 & P3 ---> P2 v Q1 v Q2. However, in order to retain the full intention behind each rule, the following logically equivalent clausal form is used: -P1 v -(-P2) v -P3 v Q1 v Q2.

Set of Support

With forward chaining, the path of inferencing can be directed by a set of clauses (which are facts) called the Set of Support. One clause, the "current clause", is removed from the Set of Support and a literal from the current clause is matched against the literals in the antecedent of each rule. If the literal of the current clause can satisfy a literal in the antecedent of a rule, then an attempt is made to satisfy the remaining literals in the antecedent. If the antecedent can be completely satisfied then the consequence is generated and placed in the Set of Support. If the current clause has length greater than one, then each literal in the clause is taken sequentially from left to right. After all the literals in the current clause have been matched against each rule, the current clause is removed from the Set of Support. Thus, this process is repeated until the Set of Support is exhausted (i.e. empty).

If one views the Set of Support as a stack, then the manipulation of this stack may represent a number a various search strategies. If each newly generated fact is placed on top of the Set of Support (i.e. the stack is First In Last Out), then a depth first search results. However, if each newly generated fact is placed on the bottom (i.e. the stack is First In First Out), then the Set of Support implements a breadth first search. In addition to these two standard search strategies, a number of other strategies may be implemented which may be more effective for particular problems. For example, the length of a clause being generated may be used as a criteria for choosing the current clause from the Set of Support. Thus, the Set of Support is ordered such that the shortest clauses are always on top. This helps to control looping and to prune branches from the tree in which clauses may simply expand themselves infinitely. Often even more desirable, each clause generated may be weighted either by a predefined weight for each predicate or interactively. This later approach can be very effective because (human) intelligence comes to bear on the direction of reasoning being taken.

As a short example describing the Set of Support and its implementation of search strategies, consider the following problem. In order to minimize boolean functions which are in disjunctive normal form, the simple

observation that x1x2-x3x4 v -x1x2-x3x4 can be reduced to x2-x3x4 is often used (note that in this disjunction x1 and -x1 offset each other and the truth value of this disjunction is determined by x2-x3x4 v x2-x3x4). In order to minimize functions of four variables, the facts describing all the possible reductions for four variables and for three variables are needed in order to begin. They may be represented by the following predicates:

```
Reduce4(x1x2x3x4  x1x2x3-x4  x1x2x3)
Reduce4(x1x2x3x4  x1x2-x3x4  x1x2x4)
   .      .         .         .
   .      .         .         .
Reduce4(x1-x2-x3-x4  -x1-x2-x3-x4  -x2-x3-x4)
Reduce3(x1x2x3  x1x2-x3  x1x2)
Reduce3(x1x2x3  x1-x2x3  x1x3)
   .      .        .      .
   .      .        .      .
Reduce3(x1-x2-x4  -x1-x2-x4  -x2-x4)
```

Only the following two rules are needed:

```
1:  Var4(X)  &  Var4(Y) &  Reduce4(X,Y,Z)  --->
    Var3(Z)

2:  Var3(X)  &  Var3(Y) &  Reduce3(X,Y,Z)  --->
    Var2(Z)
```

Taking the function:

```
F = -x1x2x3-x4  v  x1-x2x3-x4  v  x1x2x3-x4
    v x1x2-x3-x4  v  -x1x2-x3-x4
```

initially, the Set of Support is:

```
3: Var4(-x1x2x3-x4)
4: Var4(x1-x2x3-x4)
5: Var4(x1x2x3-x4)
6: Var4(x1x2-x3-x4)
7: Var4(-x1x2-x3-x4)
```

Figure 2 gives a snap shot view of the Set of Support. At each step, the current clause being used to make deductions is removed from the Set of Support and all facts which were deduced using this clause are placed on the bottom of the stack. The numbers to the right of the deduced facts represent which facts were used to make the reduction.

At this point, the remaining clauses will simply be popped from the Set of Support with no new facts being deduced and the Set of Support will thus exhaust itself. From the results, if we use the number of a fact, which represents a term in the equation (rather then the term itself), it is easy to see that:

```
F   = #3 v #4 v #5 v #6 v #7

#14 = #9 v #11

#9  = #3 v #7
#11 = #5 v #6
```

hence,

```
#14 = #3 v #5 v #6 v #7   and
F   = #14 v #4
```

therefore,

```
F = -x1x2x3-x4  v  x1-x2x3-x4  v  x1x2x3-x4  v
    x1x2-x3-x4 v -x1x2-x3-x4

  = x2-x4 v x1-x2x3-x4.
```

The preceding example exhibits a breadth-first search of all possible reductions. That is, all reductions to three variables are made first and then, after this, all reductions to two variables are made. However, if the deduced

```
1) Current: Var4(-x1x2x3-x4)

      SoS: Var4(x1-x2x3-x4)
           Var4(x1x2x3-x4)
           Var4(x1x2-x3-x4)
           Var4(-x1x2-x3-x4)
        8: Var3(x2x3-x4)        3,5
        9: Var3(-x1x2-x4)       3,7

2) Current: Var4(x1-x2x3-x4)

      Sos: Var4(x1x2x3-x4)
           Var4(x1x2-x3-x4)
           Var4(-x1x2-x3-x4)
           Var3(x2x3-x4)
           Var3(-x1x2-x4)
       10: Var3(x1x3-x4)      4,5

3) Current: Var4(x1x2x3-x4)

      SoS: Var4(x1x2-x3-x4)
           Var4(-x1x2-x3-x4)
           Var3(x2x3-x4)
           Var3(-x1x2-x4)
           Var3(x1x3-x4)
       11: Var3(x1x2-x4)       5,6

4) Current: Var4(x1x2-x3-x4)

      SoS: Var4(-x1x2-x3-x4)
           Var3(x2x3-x4)
           Var3(-x1x2-x4)
           Var3(x1x3-x4)
           Var3(x1x2-x4)
       12: Var3(x2-x3-x4)      6,7

5) Current: Var4(-x1x2-x3-x4)

      Sos: Var3(x2x3-x4)
           Var3(-x1x2-x4)
           Var3(x1x3-x4)
           Var3(x1x2-x4)
           Var3(x2-x3-x4)

6) Current: Var3(x2x3-x4)

      SoS: Var3(-x1x2-x4)
           Var3(x1x3-x4)
           Var3(x1x2-x4)
           Var3(x2-x3-x4)
       13: Var2(x2-x4)      8,12

7) Current: Var3(-x1x2-x4)

      SoS: Var3(x1x3-x4)
           Var3(x1x2-x4)
           Var3(x2-x3-x4)
           Var2(x2-x4)
       14: Var2(x2-x4)      8,11

8) Current: Var3(x1x3-x4)

      SoS: Var3(x1x2-x4)
           Var3(x2-x3-x4)
           Var2(x2-x4)
           Var2(x2-x4).
```

Fig 2 : Set of Support with breadth first search

facts were placed on top of the stack rather than on the bottom, a depth first search results. Then with the first reduction to three variables, (and with every reduction thereafter) an attempt is made to reduce this new fact immediately. Although this example is rather trivial, it is possible to perceive how one can control the path of inferencing by manipulating the Set of Support in a number of different ways. Weighting each clause as it is placed in the Set of Support and then popping the clause with the lowest weight can also be very effective. Finally, due to the rapid growth of "long" facts, it is also advantageous to place the clauses of the shortest length on top of the Set of Support.

Variable Free Resolution and Subsumption

With clausal form, the generation of facts requires special attention. This is due to facts which have a length greater than one. These facts are generated in one of two ways. First, given the rule P1 & ... & Pn ---> Q1 v ...v Qn is generated. Second, given the rule P1 &...& Pn ---> Q1 and the facts P2,...,Pn and the fact P1 v S1 v...v Sn, the fact Q1 v S1 v...v Sn is generated. Normally, in an automatic theorem prover the goal of the system is to reach the null clause. However, if one makes the goal of the system to attempt to determine the truth values for each predicate, then the clause of the shortest length is most desirable. In order to keep the facts as concise and short as possible, with each deduction two final steps are taken with the deduced fact - variable free resolution and subsumption.

By definition, facts are clauses which are variable free. Hence, we may use variable free resolution and from the facts themselves produce new facts without the use of a rule. In general, given the two facts L1 v C1 and -L1 v C2 where L1 is a literal and C1 and C2 are the remaining disjunctions of literals (i.e. clauses) within their respective facts, then the fact C1 v C2* may be deduced. There are two instances in which variable free resolution is advantageously applied to facts. The first case is given one fact which has length greater than one and a second fact of length one which can be resolved with the first fact. For example, if the following two facts exist P1 v R1 v...v Rn and -P1, then the fact R1 v...v Rn is generated from them. The second case is given two facts, say, P1 v C1 and -P1 v C1, where P1 and -P1 are literals and C1 is the remaining clause, then the fact C1 v C1 which equals C1 is generated. For example, from the two facts, P1 v Q1 and -P1 v Q1, the fact Q1 is resolved. Unfortunately, other applications of variable free resolution to facts do not tend to yield useful facts. To see this, take P1 v C1 and -P1 v C2 and resolve the fact C1 v C2. Any further deductions with the fact C1 v C2 using the rules may also be generated using one of the two initial facts (e.g. for any deductions made using a literal in C1, simply take the deduced fact and replace C2 by P1. Likewise, for all facts deduced using C2, simply replace C1 with -P1). Furthermore, any variable free resolutions using the fact C1 v C2 or any of its descendant may also be reached by using the to initial facts. For example, say -C1 exists, then this may be used with P1 v C1 to resolve the fact P1. Then this fact is used with -P1 v C2 to resolve C2, which may be directly resolved using the facts -C1 and C1 v C2.

The generation of facts as clauses also produces inefficiencies due to redundant or less specific facts. The use of subsumption helps control the rapid growth of these facts. Again, if the goal of the system is to find the truth values of each predicate then facts which are less specific are subsumed (i.e. simply discarded). This is possible because all the descendants of these facts can be reached by shorter more specific facts. Generally, any fact L1 v L2 V ... V Ln subsumes another fact, M1 v M2 v ... v Mr if n <= r and for i = 1,2,...,n Li = Mi. For example, if the two facts L1 and L1 v A1 v B1 both exist then L1 will subsume L1 v A1 v B1. That is, for any new information to be deduced about the truth values of A1 or B1, the fact -L1 must exist in order to first resolve the clause L1 v A1 v B1 to the clause A1 v B1. However, this is impossible without having a contradiction in the system. Likewise, all new facts deduced from L1 v A1 v B1 using variable free resolution with either literal A1 or literal B1 will contain the literal L1 and hence, will also be useless. Finally, all deductions using rules will result in one of two cases. If either literal A1 or B1 is used to resolve the antecedent of a rule, then consequence which will be generated will also contain the literal L1. Second, if the rule L1 ---> C1 v...v D1 exists, then the fact A1 v B1 v C1 v...v D1 may be deduced using the fact L1 v A1 v B1. However, L1 may also be used directly to deduce C1 v...v D1, which in turn subsumes A1 v B1 v C1 v... v D1. Hence, the system simply discards L1 v A1 v B1.

Contradiction Clause

Often, if a solution is reached, it is desirable to stop the deduction process even though the Set of Support has not been exhausted. To do this a special clause called the contradiction clause is accepted by the system. If a contradiction generated by the system is acceptable, then the system stops. Generally, if the solution to a problem consists of a conjunction of literals, say, P1 &...& Pn, then the contradiction is entered as -P1 &...& -Pn. This is logically equivalent to the clause P1 v...v Pn and may be interpreted as the rule -P1 &...& -Pn ---> which, when satisfied, generates the null clause (which represents the contradiction). As there may be a set of solutions (all of which are not necessarily acceptable), upon reaching a contradiction, it is best to query where the current contradiction is satisfatory. If not, the system continues to attempt to generate other contradictions.

Conclusion

A system based on Predicate Logic, using forward chaining, can be designed for general problem solving. Such a system requires methods for controlling inefficiencies which arise when using clausal form. Automatic theorem provers, developed to verify mathematical proofs, offer various structures for improving efficiency and

* To see this remember that P1 & ... & Pn ---> Q1 v...v Qn is logically equivalent to -P1 v...v -Pn v Q1 v...v Qn. Then, L1 v C1 is equivalent to -C1 ---> L1 and -L1 v C2' is equivalent to L1 ---> C2. From this -C1 ---> C2 which is equivalent to C1 v C2.

insuring sound reasoning. Such systems, based on forward chaining offer great flexibility in searching strategies as well as more general information when facts are generated in clausal form.

References

[Bun83], A. Bundy, The Computer Modelling of Mathematical Reasoning; Academic Press; London, 1983.

L. Wos, R. Overbeek, E. Lusk, J. Boyle, Automated Reasoning; Prentice-Hall, Inc.; New Jersy, 1985

A C K N O W L E D G E M E N T

Mr. John David Freyder is leaving Ljubljana where he lived in the last year. The Editor-in-Chief of Informatica has to say that John's help in reviewing, correcting, and translating of English texts was extremely meticulous and precious. Additionally to this, John was always prepared for hours of intense conversation in which he revealed many secrets of the English idiom to authors of Informatica.

We wish and hope that John in his native Chicago will find as fruitful, productive, and pleasant relations as he has found in Ljubljana. We shall seriously miss his knowledge, intelligence, patience, and friendness. We hope very much that it will be possible to keep the continuation of our collaboration on force.

Lojze Vogel
Institut Jožef Stefan
Ljubljana

UDK 681.324

Članek predstavlja vmesnik SPI-11. Razvit je bil za potrebe mikroračunalnika PMP-11 z namenom, da se povečajo njegove komunikacijske zmogljivosti. Opisana je izvirna aparaturna zgradba in prikazana modularna zasnova vmesnika. Programski in aparaturni združljivosti z že razvitimi moduli PMP-11 je bila v razvojni fazi namenjena posebna pozornost.

COMMUNICATION INTERFACE SPI-11: In this paper the communication interface SPI-11 is described. It was developed for microprocessor PMP-11 to enhance its communication capabilities. Original design and modular approach of the interface is presented. Special attention was paid to software and hardware compatibility with already developed PMP-11 moduls.

1 UVOD

Serijski in paralelni komunikacijski vmesnik SPI-11 predstavlja enega od modulov, ki dopolnjujejo mikroračunalnik PMP-11 (enokartični mikroračunalnik na osnovi Digital-ovega mikroprocesorja DCT-11, ki je programsko v celoti skladen s sistemi PDP-11 in LSI-11 ter domačo družino računalnikov Iskra Delta pod operacijskim sistemom RT-11). Vsi uporabiniki mikroračunalnika PMP-11, ki pri svojem delu potrebujejo večje komunikacijske zmogljivosti (v sistemu brez SPI-11 sta na voljo dve asinhroni serijski liniji RS-232 s hitrostjo 19200 baudov), lahko z dodanim vmesnikom pridobijo še nadaljnih sedem serijskih linij. Pri tem lahko dva od serijskih kanalov direktno krmilita modem, nadaljna dva pa podpirata komunikacijo preko 20mA tokovnih zank. Potrebo po paralelnem komuniciranju s poljubno uporabnikovo periferijo pokriva 24-signalni paralelni vmesnik, ki je sestavni del novega komunikacijskega modula.

Odlika SPI-11 vmesnika je njegova modularna zasnova, ki omogoča enostavno paralelno povezovanje nadaljnih SPI-11, če se pokaže potreba po razširitvi komunikacijskih zahtev. V maksimalni konfiguraciji je dovoljena paralelna povezava štirih SPI-11 na en mikroračunalnik. Ta povezava je v celoti aparaturno in programsko skladna s krmilnikom trdega (Winchester) diska, ki je priključen na PMP-11 I/O vodilo na enak način kot komunikacijski vmesnik-i.

2 PREDSTAVITEV VMESNIKA

2.1 Izhodišča pri zasnovi

Standardni format tiskanine za PMP-11 (dimenzije 162 X 141 mm) bistveno določa domet komunikacijskega vmesnika. Poleg formata pa so na zasnovo vplivale še druge zahteve in omejitve. Najpomembnejša zahteva je hkratna aparaturna in programska skladnost z mikroračunalnikom in krmilnikom trdega diska. Pri tem je potrebno upoštevati, da je lahko na PMP-11 I/O vodilo paralelno priključenih več krmilnikov oziroma vmesnikov. Vmesniku SPI-11 je dodeljenih šest pomnilniških lokacij iz naslovnega prostora I/O strani. Tem naslovom so v SPI-11 dodeljeni registri, ki so dostopni procesorju. Ker mikroračunalnik PMP-11 pred vsakim vpisom zahteva tudi branje pomnilniške lokacije, mora krmilna enota aparaturno onemogočiti branje v primeru registra, ki dovoljuje samo vpisovanje (tak slučaj predstavlja krmilni register elementa 8255A). Upoštevanje podobnih zahtev in omejitev je privedlo do kompromisne rešitve, ki je narekovala izbiro željenih funkcij.

Aparaturno je serijski in paralelni komunikacijski vmesnik tako zasnovan, da je možna paralelna povezava dveh, treh, največ pa štirih vmesnikov na en mikroračunalnik PMP-11. Na ta način je dana možnost, da se osem komunikacijskih kanalov (7 serijskih linij in en paralelni vmesnik) razširi na 16, 24 oziroma 32 kanalov. V primeru uporabe več SPI-11 je potrebno na vsaki tiskanini nastaviti različno kodo modula. Modul je aktiven samo v slučaju, kadar se njegova fiksno izbrana koda ujema z naslovom, ki ga pošlje mikroračunalnik v trenutku, ko izbira komunikacijski kanal.

Na SPI-11 sta zastopana samo dva osnovna komunikacijska elementa: serijska linija RS-232 in paralelni vmesnik. Na eno tiskanino je bilo moč razporediti največ sedem serijskih linij in en 24-signalni paralelni vmesnik. Razpoložljiv prostor je namenjen še dvema 20mA tokovnima zankama in vezju, ki podpira modemske signale za dve serijski liniji.

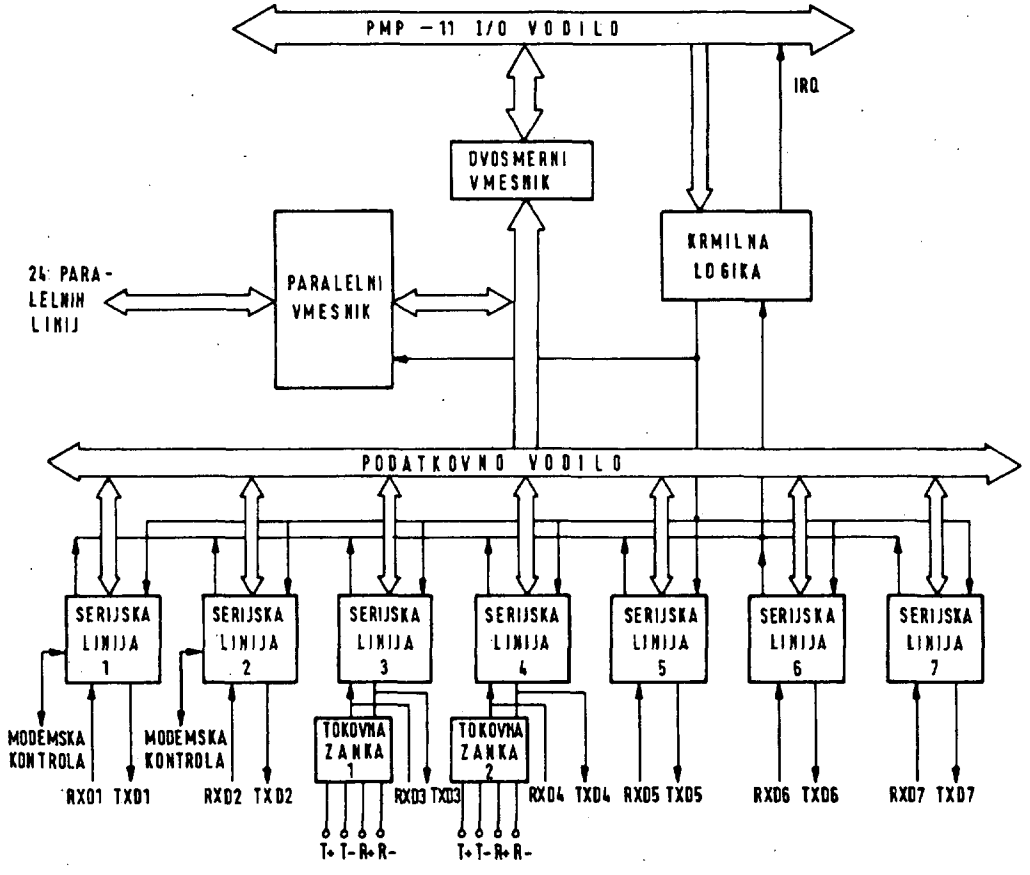
2.2 Krmilna enota

Vse potrebne krmilne signale, ki skrbijo za funkcijski nadzor, generira krmilna enota. Le-ta je zaradi dobre izkoriščenosti razpoložljivega prostora na tiskanini realizirana z dvema PAL (Programmable Array Logic) elementoma , ki generirata signale za nadzor pretoka podatkov na ta način, da se v skladu z vrsto ukaza aktivirajo odgovarjajoče prenosne poti med mikroračunalnikom in komunikacijskim vmesnikom. Poleg PAL elementov sestavljajo krmilno enoto še naslednje komponente: register za izbor kanala skupaj z naslovnim dekoderjem, prekinitvena logika, register prekinitvenih mask in naslovni register za izbor kanala. Krmilna enota sprejema mikroračunalnikove ukaze, jih razpoznava in aktivira ustrezne krmilne elemente, ki poskrbijo, da se željena funkcija pravilno realizira.

2.3 Serijska linija

Osnovni gradnik serijske linije je vezje USART - 8251A (Universal Synchronious/Asynchronious Receiver/Transmitter), ki omogoča serijsko komunikacijo. V primeru oddaje poljubnega znaka pošlje mikroračunalnik 8-bitni podatek izbranemu USART elementu, ki ga pretvori v serijski podatkovni niz za oddajo. Hkrati lahko sprejema serijski podatkovni niz in ga pretvarja v 8-bitni paralelni podatek, ki je mikroračunalniku na voljo za branje v podatkovnem registru. USART obvešča procesor o sprejemu novega znaka in o tem, da je pripravljen za oddajo naslednjega zanaka.

Uporabniku je na voljo sedem serijskih linij in pri vsaki je možno izbrati različno hitrost prenosa podatkov. Na voljo so naslednje hitrosti: 300, 600, 1200, 2400, 4800 in 9600 baudov. SPI-11 podpira full-duplex prenos na vseh serijskih linijah. Dve serijski liniji (kanala 0 in 1) omogočata tudi komunikacijo preko modema. Ta dva kanala podpirata standardne modemske signale (Data Set Ready, Data Terminal Ready, Request to Send, Clear to Send).



Shema SPI-11

Izbor kanala, s katerim želi mikroračunalnik vzpostaviti zvezo, se izvede na ta način, da sistem pošlje komunikacijskemu vmesniku naslov željenega kanala, ki ga vmesnik shrani v naslovni register, dekodira in aktivira izbrano linijo, po kateri vzpostavi komunikacijo s priključeno periferno napravo. Povezava z obstoječim kanalom se prekine, ko mikroračunalnik pošlje krmilni enoti nov naslov. Binaren naslov kanala, s katerim želi procesor komunicirati, je sestavljen iz dveh delov, tako da trije najmanj pomembni biti predstavljajo naslov kanala znotraj izbranega SPI-11. Dva najbolj pomembna bita pa izbereta enega izmed štirih komunikacijskih vmesnikov, ki so lahko hkrati priključeni na mikroračunalnikovo vhodno/izhodno vodilo.

2.4 Prekinitvene zahteve

Na vhodno/izhodnem vodilu mikroračunalnika je ena sama linija IRQ, po kateri periferija javlja procesorju potrebo po takojšnjem servisiranju prekinitvene zahteve. Na to linijo je priključena tudi skupna prekinitvena zahteva, kot rezultat združitve posameznih zahtev nastalih na serijskih kanalih SPI-11 vmesnika.

Na vmesniku je realizirana tudi funkcija maskiranja. Uporabnik ima na ta način možnost, da v določenem časovnem intervalu dovoljuje oziroma onemogoča servisiranje trenutnih prekinitvenih zahtev, ki so posledica dogodkov na serijskih linijah. Vsaka serijska linija ima v registru prekinitvenih mask svoj maskirni bit. Prekinitvena zahteva nekega kanala je omogočena samo v primeru, ko je njen pripadajoči maskirni bit na logični enici. Čeprav ima vsaka serijska linija zase interno možnost maskiranja, je prednost registerskega maskiranja v tem, da je časovno hitrejše, saj je izvedeno z enim samim mikroračunalnikovim ukazom (z vpisom v register prekinitvenih mask).

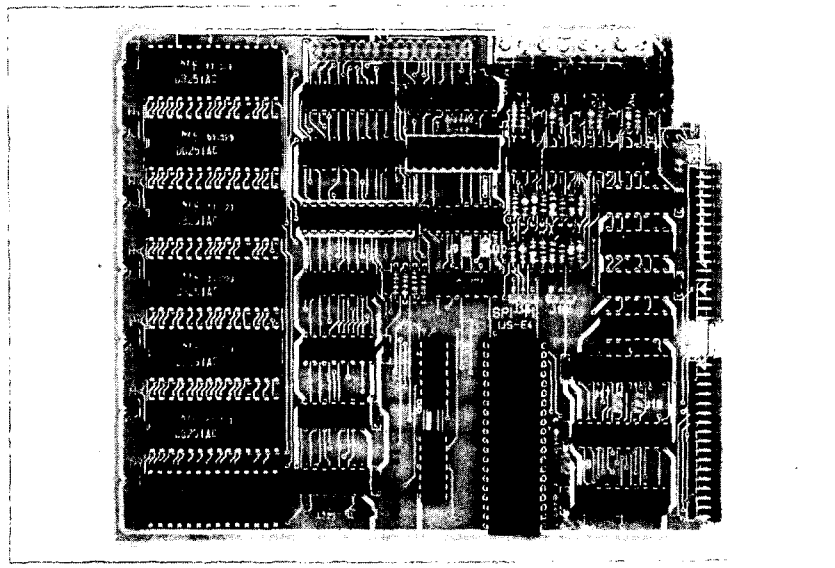
Prekinitveno logiko sestavlja poleg registra prekinitvenih mask še PAL vezje, ki združuje vse parcialne zahteve v skupno prekinitveno zahtevo. Tako preprost prekinitveni mehanizem je bil izbran z namenom, da zavzame čim manj potrebnega prostora na tiskanini, kar pa ima za posledico več potrebnih mikroračunalnikovi- ukazov ob servisiranju prekinitvene zahteve, saj je potrebno z branjem statusnih registrov v USART-ih ugotoviti na kateri liniji je prekinitev nastala.

2.5 Paralelni vmesnik

Poleg opisanih serijskih povezav omogoča SPI-11 vmesnik tudi paralelno komuniciranje z okolico. Paralelni vmesnik temelji na elementu 8255A (Programmable Peripheral Interface), ki podpira različne načine delovanja. Uporabnik izbere izmed možnih povezav tisto, ki najbolj ustreza njegovemu konkretnemu problemu. Željeno konfiguracijo vmesnika definira ustrezna programska podpora in nastavitev mostičkov, ki določa vhodno oziroma izhodno smer prenosa podatkov. Na procesor je paralelni vmesnik vezan preko dvosmernega 8-bitnega podatkovnega vodila in povezuje periferijo s 24 paralelnimi signali, od katerih je procesorju hkrati dostopna le ena skupina po osem signalov.

3 ZAKLJUCEK

V našem centru za načrtovanje tiskanih vezij je bila na osnovi logičnih shem razvita prototipna tiskanina, ki je v fazi testiranja doživela še par manjših popravkov. Hkrati ob delu je nastajala potrebna dokumentacija. Na voljo je delovno poročilo z naslovom Serijski in paralelni komunikacijski vmesnik SPI-11. Navodila za uporabnika Opis mostičkov na tiskanini SPI-11 predstavljajo funkcije, način konfiguriranja in opcije vmesnika, ki jih lahko izbiramo z ustrezno nastavitvijo mostičkov. Postopek oživljanja tiskanine SPI-11 je dokumentacija, ki služi tehniku kot pripomoček pri generiranju novih SPI-11. V ta namen je razvita tudi ustrezna Testna programska podpora za SPI-11 (avtor Igor Ozimek, Institut Jožef Stefan). Doslej je bilo uspešno instaliranih že več komunikacijskih vmesnikov pri različnih uporabnikih mikroračunalnika PMP-11.



Fotografski posnetek vmesnika

MODELIRANJE INFORMACIJSKIH SISTEMOV Z UPOŠTEVANJEM ELEMENTOV DINAMIKE REALNEGA POJAVA

INFORMATICA 3/87

UDK 519.863

Brane Semulič
Cinkarna Celje

Povzetek

Članek opisuje predlog pristopa k snovanju računalniško podprtih informacijskih sistemov, ki naj bi rešil probleme konsistentnosti sistemov, ki nastopajo zaradi tega, ker obstoječi pristopi običajno ločijo projektiranje baze podatkov od projektiranja programov, ki te podatke uporabljajo.

Summary

In this paper an approach to computer supported information system design is presented. With described approach we can solve the problems of system consistency which occur in case of usual design when in separated ways is approached to the data base design and to the programs design.

1 U V O D

Predlagan pristop se nanaša na izgradnjo tistega dela poslovnih informacijskih sistemov, ki so determinirani in jih je možno detajlno strukturirati. Pri tem izhajamo iz dognanj teorije in prakse baz podatkov.

Pri snovanju večjih kompleksnih programskih proizvodov se srečujemo s problemi, ki se nanašajo na konsistentnost celotnega sistema, ker obstoječi pristopi v veliki meri ločijo projektiranje baze podatkov od projektiranja programov, ki te podatke uporabljajo, kar pomeni, da ne upoštevajo časovne povezanosti med samimi programi, kakor tudi med programi in podatki. Rezultat vsega tega se kaže v nerešenih problemih sinhronizacije.
Razrešitev opisanih problemov lahko dosežemo z uporabo pristopa, ki ga opredeljujeta sledeči fazi (6):

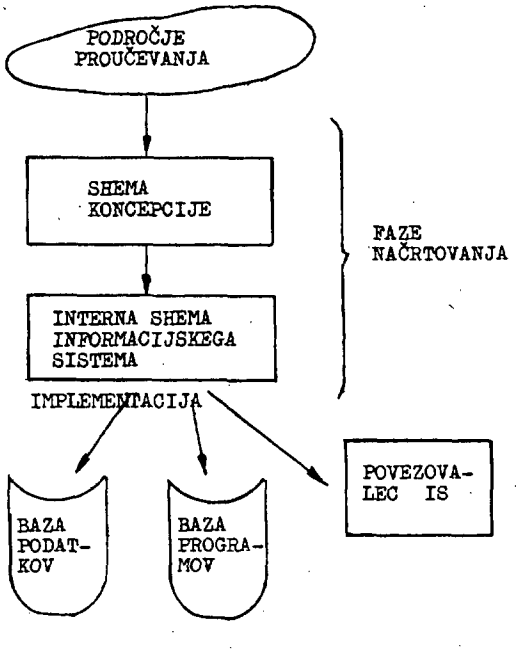
- izdelava sheme koncepcije informacijskega sistema in
- izdelava interne sheme informacijskega sistema.

Rezultat prve faze je "SHEMA KONCEPCIJE INFORMACIJSKEGA SISTEMA", ki zagotavlja:

- celovito,
- konsistentno,
- neredundantno in
- jasno

semantično predstavitev proučevanega pojava.

Naloge in prednosti sheme koncepcije informacijskega sistema so enake kot pri shemi in podshemi baze podatkov, vendar s to razliko, da daje navedena shema kompleksnejši prikaz obravnavanega realnega pojava, ker upošteva tudi element časa. V naslednji fazi, ki obsega izdelavo "interne sheme informacijskega sistema", pa dopolnimo rešitve, ki smo jih dobili z izvedbo prve faze, s tehničnimi karakteristikami in parametri, ki smo jih prej izpustili. Rezultat interne sheme informacijskega sistema nato uporabimo za implementacijo programskega proizvoda, ki ga v tem kontekstu zaokrožuje struktura podatkov, ki je zbrana v bazi podatkov, zbirka programov in ustreznih transakcij v bazi programov ter zbirka kontrolnih ukazov, ki jih aktiviramo s pomočjo povezovalca informacijskega sistema, ki sproža programe in transakcije med posameznimi podatki. Na sliki 1 imamo ilustrativen prikaz postopka izgradnje tako zasnovanega programskega proizvoda.



Slika 1

Vlogo povezovalca informacijskega sistema lahko prevzamejo različni akterji, kot so na primer: ročni postopki, za to posebej izdelani programi, kombinacije obeh omenjenih, sistem za upravljanje z bazo podatkov (DBMS) ipd.

V nadaljevanju se bomo omejili na prikaz prve faze predlaganega pristopa, ki obsega izdelavo sheme koncepcije informacijskega sistema.

2 IZDELAVA SHEME KONCEPCIJE INFORMACIJSKEGA SISTEMA

2.1 Izhodišča izdelave sheme koncepcije informacijskega sistema

Pri izdelavi sheme koncepcije informacijskega sistema veljajo enake zakonitosti, kot pri izdelavi logične sheme baze podatkov, vendar s to razliko, da je shema koncepcije delovanja informacijskega sistema obravnavana s širšega vidika. Pri izdelavi omenjene sheme upoštevamo vse vidike realnega pojava in ne samo statični vidik organizacije kot je to slučaj pri bazi podatkov. Tukaj imamo v mislih predvsem transakcije, ki vsakokrat povzročijo vzpostavitev novega konsistentnega stanja znotraj sistema baze podatkov.
Zaradi opisanih lastnosti transakcij jim pravimo, da predstavljajo "vedenjske enote" baze podatkov. S transakcijami generiramo torej novo stanje baze podatkov, ki ustreza dejanskemu stanju evolucije realnega pojava.

Pri opazovanju realnega pojava zasledimo, da le ta reagira na vsak dogodek, s tem, da s pomočjo izvedbe določenih aktivnosti vzpostavi novo stanje. Podobno situacijo imamo pri sistemih podatkov, ko s transakcijami vzpostavimo novo stanje, ki zrcali stanje zasledovanega realnega pojava.

Da se izognemo nekoordiniranosti delovanja sistema baze podatkov glede na delovanje realnega pojava, morajo biti transakcije, s katerimi spreminjamo stanje baze podatkov, preslikava aktivnosti zasledovanja realnega pojava.

Na osnovi zapisanega lahko podamo sledeči ugotovitvi, in sicer da (5):

- s podatki ponazarjamo podatkovne pojme (entitete) realnega pojava in da
- s transakcijami ponazarjamo vedenje proučevanega realnega pojava.

Tako razširjeno gledanje nam je osnova za pristop k izdelavi sheme koncepcije informacijskega sistema, ki ga izdelamo brez upoštevanja kakršnihkoli tehničnih parametrov, ki jih je potrebno upoštevati za implementacijo modela na konkretnem računalniškem sistemu.

V našem gledanju modela koncepcije informacijskega sistema imamo torej formalizirano integracijo treh vidikov, in sicer:

- statične strukture elementov modela informacijskega sistema,
- transakcije med elementi modela informacijskega sistema in
- časovni vidik relacij med posameznimi elementi modela informacijskega sistema.

Če proučujemo realni pojav z opisanih treh vidikov, pridemo do sledečih ugotovitev (4):

- da obstajajo tri osnovne kategorije pojava, ki jih je možno opisati glede na njihove lastnosti kot:
  - objekte
  - dogodke in
  - operacije.
- da je možno dinamične odnose med naštetimi kategorijami izraziti s POVEZAVAMI, ki jih glede na vrsto povezave delimo na:
  - modifikacijo (povezava: OPERACIJA-OBJEKT)
  - spoznanje (povezava: OBJEKT-DOGODEK)
  - sprožitev (povezava: DOGODEK-OPERACIJA)

V nadaljevanju si bomo opisane kategorije in njihove medsebojne povezave nekoliko pobliže ogledali.

2.2 Opisi kategorij realnega pojava in njihove povezave

Videli smo, da obstojajo tri osnovne kategorije realnega pojava, in sicer:

- objekt-predstavlja trajno konkretno ali abstraktno proučevano komponento organizacije, ki jo lahko detajlneje opredelimo,
- operacija-predstavlja nalogo, ki jo je potrebno realizirati v organizaciji ob določenem času,
- dogodek-predstavlja spoznanje o spremembi stanja enega ali več objektov po izvedbi operacij

Med posameznimi opisanimi kategorijami obstojajo povezave, ki jih opredelimo sledeče:

- spoznanje-je povezava, ki združuje dogodek in enega ali več objektov. Izraža spremembo stanja objektov, ki je povzročena s pojavom nekega dogodka,
- sprožanje-je povezava, ki združuje dogodek in eno ali več operacij. Pove, da določen dogodek sproži eno ali več operacij,
- modifikacija-predstavlja zvezo med operacijami in objekti. Izraža kako operacija preoblikuje objekt.

Iz opisanega je moč razbrati, da so dogodki tista kategorija, s katero opredeljujemo dinamiko realnega pojava.

2.3 Opredelitev dinamike realnega pojava

Dinamiko realnega pojava opredeljujejo dogodki, ki jih ločimo na (7):

- pasivne dogodke in
- aktivne dogodke.

Pasivni dogodki povzročijo samo novo stanje proučevanega objekta, medtem, ko aktivni dogodki sprožijo izvajanje ene ali več operacij.

Opisana dinamika realnega pojava temelji na jasnem razlikovanju med:

- stanjem,
- spremembo stanja, ki jo povzročijo pasivni dogodki in
- spremembo stanja, ki jo povzročijo aktivni dogodki.

Stanje nekega objekta je lahko trajno, medtem, ko je sprememba stanja dogodek in je trenutna. Pasivni dogodek izraža prehod iz enega stanja v drugo. Aktivni dogodek se razlikuje od pasivnega po tem, da spremembo stanja povzroči tudi izvajanje ene ali več operacij.

2.4 Formalna opredelitev realnega pojava, njegovih kategorij in njihovih lastnosti

2.4.1 Opredelitev na nivoju realnega pojava

Realen pojav, ki ga v našem primeru zaokrožuje temeljni proces poslovnega sistema, bomo skušali opredeliti na sledečih nivojih (7):

- nivoju realnega pojava
- nivoju pojmovnega modela in
- nivoju podatkovnega modela

Pri proučevanju temeljnega procesa na nivoju realnega pojava ugotovimo, da ga je možno opredeliti z množico veličin, s katerimi opišemo njegove karakteristike. Te veličine označujemo kot objekte (OB_R). Objekt (OB_R) proučevanja v temeljnem procesu poslovnega sistema je lahko "kupec", "naročilo", "proizvod" itd. Dogodki (DG_R) pa povzročajo spremembe stanja opazovanih objektov (S^{OB}_R), ki po dogodku preidejo v neko novo stanje.

2.4.2 Opredelitev na nivoju pojmovnega modela

Preslikava temeljnega procesa na nivo pojmovnega modela nam služi za prikaz našega gledanja temeljnega procesa, ki abstrahira za nas nepomembne vidike gledanja tega procesa in prikazuje samo tiste elemente in njihove medsebojne odnose, ki so za nas pomembni. S ciljem takšne opredelitve modela temeljnega procesa (TP_p) smo ugotovili, da je v osnovi sestavljen iz treh temeljnih gradnikov - kategorij in sicer: objektov (OB_p), dogodkov (DG_p) in operacij (OP_p). To lahko zapišemo takole (7):

$$TP_p = (OB_p, DG_p, OP_p) \qquad (1.1)$$

pri čemer je:

$$OB_p = \{ OB_p^i \; ; \quad i = 1,2,3,\ldots,n \} \qquad (1.2)$$

$$DG_p = \{ DG_p^i \; ; \quad i = 1,2,3,\ldots,m \} \qquad (1.3)$$

$$OP_p = \{ OP_p^k \; ; \quad k = 1,2,3,\ldots,l \} \qquad (1.4)$$

Povezave, ki nastopajo med posameznimi kategorijami, pa napišemo takole:

- modifikacija (MO_p) predstavlja preslikavo operacije (OP_p) v objekt (OP_p)

$$MO_p : OP_p \longrightarrow OB_p \qquad (1.5)$$

- spoznanje (SZ_p) predstavlja preslikavo objekta (OB_p) v dogodek (DG_p)

$$SZ_p : OB_p \longrightarrow DG_p \qquad (1.6)$$

- sprožitev (SP_p) predstavlja preslikavo dogodka (DG_p) v operacijo (OP_p)

$$SP_p : DG_p \longrightarrow OP_p \qquad (1.7)$$

Stanje (Sp) proučevanega pojava (TP_p) ugotovimo tako, da zasledujemo opisane kategorije in njihove medsebojne povezave (PV_p). Pri tem je potrebno izpostaviti, da nas zanimajo samo tista stanja, ki so dogodki. Formalno izrazimo to takole (7):

$$S_p = (TP_p, PV_p) \qquad (1.8)$$

pri čemer je:

$$TP_p = (OB_p, DG_p, OP_p) \quad \text{in} \qquad (1.9)$$

$$PV_p = (MO_p, SZ_p, SP_p) \qquad (1.10)$$

Spremembo stanja (Sp) povzroči izvedba transakcij, ki predstavljajo osnovne elemente poslovanja v temeljnem procesu poslovnega procesa. Transakcijo sproži dogodek (DG_p), ki povzroči izvajanje ene ali več operacij (OP_R), s katerimi se izvede sprememba na ustreznih objektih (OB_R). V proučevanem pojavu (TP_p) se lahko izvaja sprožanje dogodkov (DG_p):

- v kronološki odvisnosti (KO_p) ali pa
- v pogojni odvisnosti (PO_p).

Dogodek DG_p^i je v kronološki odvisnosti (KO_p) z dogodkom DG_p^i takrat, kadar dogodek DG_p^i nepogojno sproža operacije OP_p^k, ki povzroči spremembe stanja objekta OB_p^j, kar spoznamo z dogodkom DG_p^j.

Dogodek DG_p^j je v pogojni odvisnosti (PO_p) z dogodkom DG_p^i takrat, kadar dogodek DG_p^i pogojno sproži operacijo OP_p^k ali kadar sprememba stanja, povzročena z operacijo OP_p^k ne povzroči vedno dogodka DG_p^j.

2.4.3 Opredelitev na nivoju podatkovnega modela

Proučevanje realnega pojava (TP_D) z nivoja podatkovnega modela predstavlja preslikavo pojmovnega modela z vidika opredelitve in analize potrebnih podatkov. Pri proučevanju kategorij realnega pojava (OB_D, DG_D in OP_D) ugotovimo, da pripadajo oziroma, da so povezani s posameznimi podatkovnimi razredi (R_D). Značilnosti, ki združujejo posamezne kategorije realnega pojava znotraj podatkovnega razreda, lahko formalno zapišemo takole (7):

$$(\forall R_D^i)\,(\forall x)\,(K_D^n)\,(x \in K_D^n \Rightarrow x \in R_D^i \& L_{(x)})$$

Se pravi, da pri danem podatkovnem razredu (R_D^i) in opredeljeni lastnosti (L), ki je smiselna za atribute (X) podatkovnega razreda (R_D^i), eksistira kategorija (K_D^n), ki ima za atribute natanko tiste atribute podatkovnega razreda (R_D^i), ki imajo lastnosti (L).

V našem primeru imenujemo podatkovni model "shema koncepcije informacijskega sistema". Da bi zadostili potrebam izdelave predlagane sheme koncepcije in v formalni obliki opredelili prej opisane karakteristike in lastnosti proučevanega pojava, moramo najti ustrezen način formalnega prikazovanja. To dosežemo z (5):

- uporabo tipiziranega modela preslikav
- uvedbo tipov preslikav, s katerimi ponazorimo različne kategorije in

uvedbo dejavnika časa.

Povezavo med podatkovnimi razredi in opisanimi kategorijami dobimo tako, da:

- vsak podatkovni razred in vse vrste povezav prikažemo z eno ali več preslikavami,
- lastnosti podatkovnega razreda prikažemo z atributi preslikav in
- uporabimo za prikaz podatkovnega razreda množico tipiziranih preslikav, ki jih označimo kot:

  -R-objekt,
  -R-dogodek in
  -R-operacija (6).

Navedene tipe preslikav izrazimo v tretji normalizirani obliki grupiranja podatkov v stavke.

Za potrebe dekompozicije podatkovnega razreda na elementarne podatkovne zapise se poslužujemo tipov preslikav. V nadaljevanju bomo podali kratke definicije posameznih tipov omenjenih preslikav:

- preslikava "R-objekt" ($R\text{-}OB_D$) je permanentna preslikava, kjer je vsak atribut v permanentni odvisnosti od ključa preslikave. "R-objekt" predstavlja elementarni položaj opazovanega podatkovnega razreda v sistemu. Ta preslikava nam omogoča, da prikažemo elemente proučevanega pojava in njihovo strukturo. Še bolj precizno povedano, ta preslikava nam omogoča časovno konsistentni vidik opazovanega realnega pojava.

  Permanentna odvisnost med dvema atributoma A in B je elementarna, direktna in kanonična odvisnost, kjer je vsak dogodek a, ki pripada atributu A, odvisen od dogodka b, ki pripada atributu B. Dogodka a in b imata enako življensko dobo. Zapisano trditev lahko izrazimo tudi takole (7):

$$A \dashrightarrow B = \left\{ (a,b):\ (\forall a)\ (\forall b)\ (a \in A \wedge b \in B \wedge a \wedge b \mid_{(a,b)}) \right\} \qquad (1.12)$$

  Atribut A implicira atribut B, pri čemer je a dogodek atributa A in b dogodek atributa B. Za vsak dogodek a in b velja, da sta elementa množice dogodkov atributa A oziroma B in da imata lastnost L.

- preslikava "R-operacija" ($R\text{-}OP_D$) je permanentna preslikava, ki se nanaša na določen podatkovni razred in je v tesni povezanosti s preslikavo "R-objekt" ($R\text{-}OB_D$).

  S to preslikavo opišemo spremembo stanja (TIP SPREMEMBE), ki se izvrši na določenem "R-objektu".

Normaliziran zapis relacije "R-operacija" mora odgovarjati naslednjim zahtevam (6):

$$R\text{-}OP_D \dashrightarrow R\text{-}OB_D \qquad (1.13)$$

$$R\text{-}OP_D \dashrightarrow \text{TIP SPREMEMBE} \qquad (1.14)$$

$$R\text{-}OP_D \dashrightarrow \text{TEKST OPERACIJE} \qquad (1.15)$$

Spremembe se lahko pojavijo kot kreacija, destrukcija ali modifikacija. S tekstom operacije je mišljen kratek opis vsebine operacije.

Iz zapisa (1.13) vidimo, da se R-operacija nanaša na točno določen R-objekt in da z njo točno opredelimo, s kakšno vrsto stanja imamo opraviti (1.14 in 1.15).

- preslikava "R-dogodek" ($R\text{-}DG_D$) je permanentna preslikava. Njena normalizacija mora zadostiti sledečim pogojem:
  - da se nanaša na točno določen R-objekt (1.16)
  - da identificira nastalo spremembo izmed možnih tipov sprememb, ki smo jih opisali že pri opredelitvi preslikave "R-operacija" (1.17)
  - da opiše nastalo spremembo (1.18) in
  - da identificira operacije, ki se sprožijo z opisanim dogodkom (1.19)

Navedene pogoje je možno zapisati tudi sledeče (6):

$$R\text{-}DG_D \dashrightarrow R\text{-}OB_D \qquad (1.16)$$

$$R\text{-}DG_D \dashrightarrow \text{TIP SPREMEMBE} \qquad (1.17)$$

$$R\text{-}DG_D \dashrightarrow \text{OPREDELITEV SPREMEMBE} \qquad (1.18)$$

$$R\text{-}DG_D \dashrightarrow R\text{-}OP_D \qquad (1.19)$$

Sprožanje operacije, ki je povzroči dogodek, je lahko "pogojeno" ali "iterativno". V prvem primeru se odgovarjajoča operacija sproži samo v slučaju, ko so izpolnjeni določeni pogoji, ki jih poprej definiramo. Pri iterativnem sprožanju pa se sproži izvajanje operacije ob vsakem pojavu ustreznega dogodka.

2.5 Statični in dinamični vidik sheme koncepcije informacijskega sistema

V predhodnih izvajanjih smo prikazali izhodišča in osnove za pristop k izdelavi sheme koncepcije informacijskega sistema. Temeljni proces poslovnega sistema smo opredelili na nivoju realnega pojava, nivoju pojmovnega modela in nivoju podatkovnega modela. V procesu snovanja informacijskega sistema nas zanima predvsem nivo podatkovnega modela, ki predstavlja nivo opredelitve predlagane sheme koncepcije informacijskega sistema.

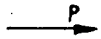
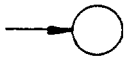
Samo shemo koncepcije informacijskega sistema delimo na dva dela, ki ju ločimo glede na vidik prikazovanja in obravnave na:

- statično podshemo (logična shema baze podatkov) in
- dinamično podshemo (shema interakcij informacijskega sistema)

Statična podshema oz. logična shema baze podatkov zajema opredelitev preslikav objektov ($R\text{-}OB_D$) realnega pojava ter njihovih lastnosti. Pri tem smatramo objekte kot podatkovne pojme, ki jih opredeljujejo ključi njihove identifikacije in atributi opisa njihovih lastnosti. S pomočjo sintakse, ki jo predvideva izdelava logične sheme baze podatkov, izrazimo povezave med tako opredeljenimi podatkovnimi pojmi.

Dinamična podshema oziroma shema interakcij informacijskega sistema pa obsega popis dogodkov ($R\text{-}DG_D$) in operacij ($R\text{-}OP_D$), s katerimi ponazarjamo dinamiko informacijskega sistema ter ilustrativen prikaz informacijskega sistema.

Za potrebe grafičnega prikaza uporabljamo sledeče simbole:

SIMBOL	POMEN
	R-objekt ($R\text{-}OB_D$)
	R-operacije ($R\text{-}OP_D$)
	R-dogodek ($R\text{-}DG_D$)
	iterativno sprožanje R-operacije ($R\text{-}OP_D$)
	pogojno sprožanje R-operacije ($R\text{-}OP_D$)
	"spoznanje", kot relacija med R-objektom ($R\text{-}OB_D$) in R-dogodkom ($R\text{-}OP_D$)
	"sprožanje", kot relacija med R-dogodkom ($R\text{-}DG_D$) in R-operacijami ($R\text{-}OP_D$)
	"modifikacija", kot relacijo med R-operacijo in R-objektom

Slika 3: Simboli, ki se uporabljajo pri izdelavi dinamične podsheme delovanja informacijskega sistema

V nadaljnjem procesu snovanja informacijskega sistema nam dinamična podshema (shema interakcij) služi kot osnova za izdelavo ustreznega programskega proizvoda, medtem, ko nam statična podshema služi kot logična shema baze podatkov, ki je izhodišče za načrtovanje ustrezne fizične zasnove baze podatkov na konkretnem računalniškem sistemu.

3 ZAKLJUČKI

Pri proučevanju realnega pojava in njegove dinamike pridemo do spoznanja, da ga opredeljujejo tri kategorije: objekti, dogodki in operacije, ki jih je možno ustrezno opredeliti na nivoju realnega pojava, nivoju pojmovnega modela in nivoju podatkovnega modela.

Če podatkovnim zapisom na nivoju podatkovnega modela, ki so normalizirani v tretji normalni obliki (3NF), dodamo še časovno omejitev, dobimo elementarne gradnike informacijskega sistema, ki jih opredelimo kot preslikave objektov ustreznih podatkovnih razredov, ki smo jih označili kot R-objekte.

Z kategorijami R-objekt, R-dogodek in R-operacija je na nivoju podatkovnega modela možno izdelati grafični prikaz delovanja informacijskega sistema. Pri tem ima takšen prikaz podobno vlogo kot logična shema baze podatkov in ga imenujemo dinamična shema koncepcije informacijskega sistema. Takšen prikaz se od logične sheme baze podatkov razlikuje v tem, da omogoča prikaz dinamike delovanja informacijskega sistema.

Opisani pristop sodi v družino sodobnih metod sistemske analize, ki temeljijo na analizi podatkov. Zaradi konsekventnih pravil, ki veljajo pri izdelavi dinamične in statične podsheme informacijskega sistema, je možno paralelno delo več strokovnjakov pri izvajanju sistemske analize kompleksnih informacijskih sistemov, ki ob zaključku dela svoje rezultate brez problemov vskladijo.

LITERATURA

1. Martin J.: Computer Data Base Organization, Prentice Hall, New Jersey, 1977

2. Prijatelj N.: Matematične strukture I, Mladinska knjiga, Ljubljana, 1971

3. Rolland C., Albin P.: Management of Real Time System Projects, Proceedings of the 7. INTERNET World Congress, Copenhagen, 1982

4. Rolland C., Benci G.: Management of Information System Projects, Proceedings of the 6. INTERNET World Congress, Garmisch-Partenkirchen, 1979

5. Rolland C., Richard C.: Transactions Modeling, Universite de Paris I, Pantheon-Sorbonne, Paris, 1982

6. Semolič B.: Magistersko delo, VEKŠ Maribor, 1986

FRAKTALI – GRAFIČNE SKRIVNOSTI RAČUNALNIŠKIH UMETNIKOV

INFORMATICA 3/87

UDK 681.3.022.06

Andrej Terčelj
Iskra Delta, Ljubljana

V prispevku so teoretično in praktično obdelani fraktali kot pripomoček pri kreiranju grafičnih slik. V neskončni množici rešitev in možnosti prikaza iterativnih matematičnih postopkov so bili nekateri med njimi obdelani in prikazani na mikroračunalniku TRIGLAV XEN-16 v višjem programskem jeziku C. Podane so smernice nadaljnega raziskovanja fraktalov in možnosti trenutne praktične uporabe ob ustrezni programski in strojni opremi.

In this article, fractals , which are used as a tool for creating graphic pictures are discused in both theoetical and practical aspects. Some of the infinite numbers of solutions and possibilities of representing iterative mathematical processes are discused and presented on the microcomputer TRIDENT XEN-16 in the programming language C. Some hints are given for future development and current possibilities for the practical applications. Problems in generating fractal geometry, using current programming technique and available hardware, are also presented.

I. UVOD

V svetu velja prepričanje, da nihče ne bo znanstveno pismen, kdor ne bo seznanjen s fraktali[2]. Fraktali so namreč začrtali in zasnovali novo pot v razmišljanju o strukturah in oblikah. Veliko pojavov se danes ne moremo matematično opisati. Z uporabo nove veje geometrije bo to mogoče razsvetliti in razrešiti. Matematična osnova fraktalne teorije je preprosta, saj temelji na ponavljanju. Omogoča nam matematično preučevanje struktur in postopkov, ki so nepravilni, neredni in neregularni, torej jih ne moremo opisati z doslej znanimi orodji. Rezultate fraktalne računalniške grafike najbolj nazorno prikažemo vizuelno. V določenih iterativnih načinih izračunavanj naletimo na osupljivo zanimive predstavitve. Tako so fraktali postali učinkovit način generiranja grafike, a dejstvo je, da brez pomoči računalnika ne gre.

II. RAZVOJ IN DEFINICIJA FRAKTALOV

Teorija fraktalov se je razvila iz praktične potrebe določiti dolžino angleške obale. Problema se je lotil leta 1967 Benoit B. Mandelbrot in ugotovil, da je dolžina morske obale nedoločljiva in odvisna od merila. Ako merilo zmanjšuješ, se dolžina povečuje in s tem tudi natančnost, to pa je mogoče nadaljevati v neskončnost. Leta 1975 je kot raziskovalec IBM Thomas J. Watson Research Centra v Yorktown Heights, New York, definiral besedo fraktal v svojem najbolj znanem delu "Les objects fractals forme, hasard et dimension". Beseda naj bi spominjala na drobljenje, lomljenje (fractus iz lat. pomeni zlomljen).

Po definiciji je fraktal oz. fraktalna geometrija matematični študij nepravilnih, večkrat kaotičnih oblik s fraktalnimi, matematično opisljivimi dimenzijami. Večji del teh oblik, struktur in procesov ohranja bistveno nespremenjeno stopnjo kompleksnosti ne glede na merilo. Drugače rečeno, to so oblike ali vzorci, ki se zdijo, kot da bi bili sestavljeni iz majhnih in še manjših oblik enega samega splošno izbranega vzorca.

Primer fraktala iz narave je drevo. Če gledaš od daleč, vidiš le krošnjo, če se približuješ opaziš rogovile, veje ,še bližje razločiš da so veje sestavljene iz vejic. Če tako nadaljuješ po nivojih vej, prideš končno do posameznih vejic, ki se v vzorcu ponavljajo. Neskončno ponavljanje preprostih vzorcev v globino v naraščajočem merilu se imenuje samopodobnost (autosimiliarity) in je osnovna in najbolj zanimiva lastnost fraktalov.

S pomočjo svoje teorije je B.B. Mandelbrot opisal poplave reke Nil, obliko oblakov in strel, Brownovo gibanje, vreme ipd. - vse z namenom vzpostaviti matematični red v "neredu" narave[7].

III. VRSTE IN OBLIKE FRAKTALOV

Fraktale delimo v dve veliki skupini glede same generacije na:

+ naključne
+ geometrične

Pri uporabi računalniške grafike je mogoče simulirati naključne fraktale in upodobiti umetne pokrajinske slike, ki jih je težko ločiti od naravnih [5], [4]. Lep primer naključnih fraktalov iz narave so meandri rek in vzorci dreves, iz celuloidne industrije je Pixar [5] do ostalih uporab v fiziki, biologiji, sociologiji in drugod [1], [3]. Na sliki 6 in 7 sta dva primera omenjene skupine. Obe sliki sta bili prikazani na barvnem monitorju in zrisani na grafični tiskalnik Fujitsu DPL24C. Listing programa prikazuje slika 8. Pot do takih in podobnih oblik je opisana v IV. poglavju. Na sliki 6 je izhodišče v točki (-2,-1.2) z dolžino stranice 1.5 , na sliki 7, ki je povečan delček slike 6, je izhodišče v točki (-1.45,0.0) in dolžino stranice 0.005.

Geometrične fraktale pa so si izmislili matematiki ob prelomu stoletja, ko so nasprotovali Newtonovi dinamiki in klasični Evklidovi geometriji (krivulja je gladka, dimenzija more biti celo število). Načrtovali so nove oblike krivulj, ki so postajale čedalje bolj vijugaste in s tem daljše, dokler niso zapolnile celotne ravnine. S tem so krivulje postale neskončno dolge in nedefinirane globine. Premica naj bi kot enodimenzionalna tvorba izgubljala enodimenzionalnost in se približevala dvodimenzionalnosti, ravnina tridimenzionalnost in tako naprej. Pot do izračuna zvijuganosti krivulje je že leta 1919 kvantitativno opisal Felix Hausdorff. Indeks je poimenoval fraktalna dimenzija krivulje (D) in je definirana z izrazom

$$N = R^{**}D$$

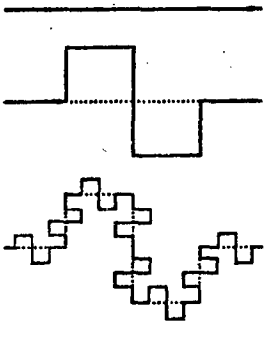
N - število odsekov pri deljenju iniciatorja

R - oblika deljenja

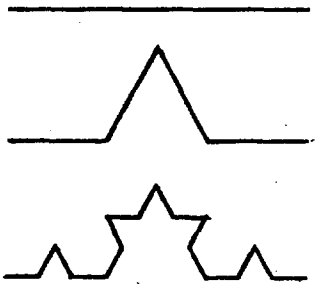
D - fraktalna dimenzija

Kako oblikujemo geometrični fraktal si poglejmo na primeru.

Na sliki 1. vidimo primer Mandelbrotove kvadratne snežinke s parametri N=8 in R=4, ter na sliki 2. Kochovo snežinko s parametri N=4 in R=3. Vsak tak geometrični fraktal se sestoji iz več nivojev. Prvi je črta, ki jo imenujemo iniciator. Izberemo si poljuben generator (nivo 2), ki iz vsakega iniciatorja generira izbrani vzorec.

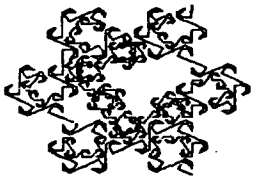


slika 1. Mandelbrotova kvadratna snežinka (D=1.5)

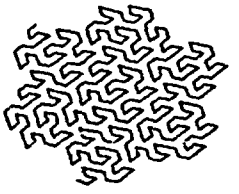


slika 2. Kochova snežinka (D=1.26)

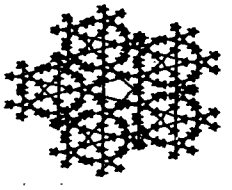
Ta proces se sukcesivno nadaljuje in ob vsaki prisotnosti iniciatorja generator ponavlja svoj vzorec v neskončno število nivojev. Po teoriji fraktalov se ponovi le končno mnogokrat, v praksi pa zadostuje že nekaj nivojev za lepo sliko. Z menjavanjem generatorja lahko ustvarimo prav nemogoče vzorce. Tri izmed njih prikazujejo slike 3,4 in 5.



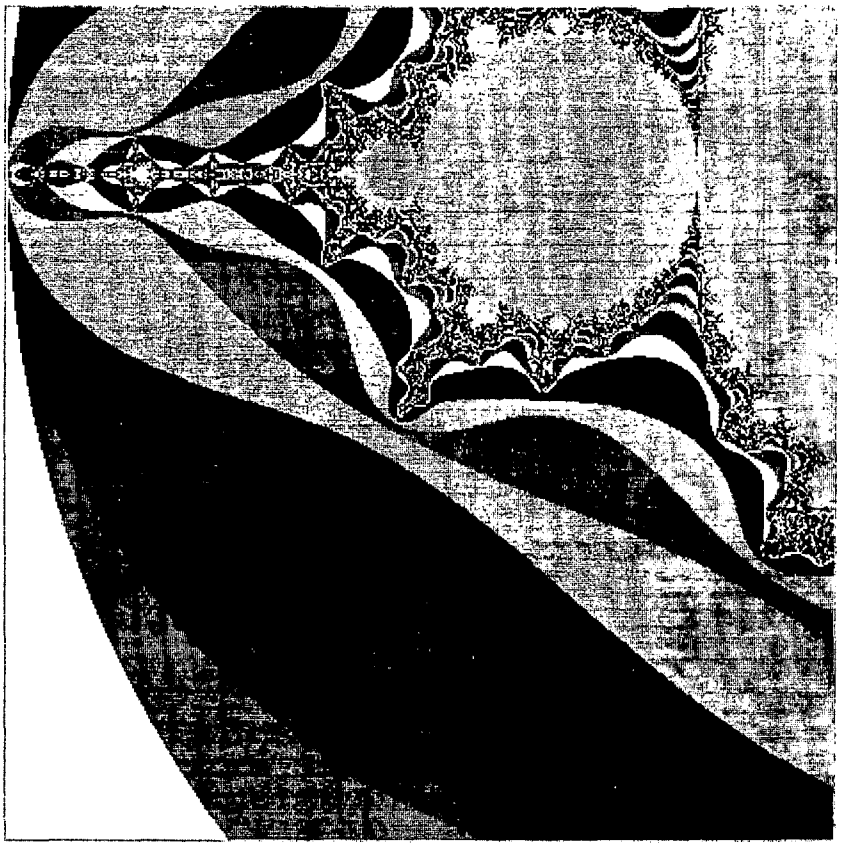
slika 3. Mandelbrotova kljukasta snežinka (D=1.8687)



slika 4. Gosperjeva zvita kača



slika 5. Abelsonov vgnezden trikotnik



slika 6. Kopija grafične slike na tiskalniku Fujitsu DPL24C

IV. OPIS ALGORITMA ZA GENERACIJO FRAKTALOV

Do oblikovanja slike obstajata dve poti:

1. Pomikamo se od točke do točke na ekranu in izračunamo dano vrednost. Prvo se postavimo v izhodišče (levi spodnji kot). Iterativno izračunamo vrednost f(z) = z**2 + C ,kjer sta z in C elementa kompleksnih stevil. Vseskozi kontroliramo vrednost z in stevec ki steje iteracije. Če stevilo iteracij preseže določeno mejo, ali z dobi vrednost 2, stevilo iteracij delimo po modulu s stevilom barv in narišemo točko v izračunani barvi. Pogoj z=2 je merilo zato, ker bo po teoriji kompleksnih stevil stevilo iteracij vodilo z v neskončnost takrat in le takrat, ko bo z dosegel vrednost 2. Opisan postopek ponavljamo za vsak element ekrana posebej. Posebej zanimiva so naslednja področja raziskave kompleksne konstante C:

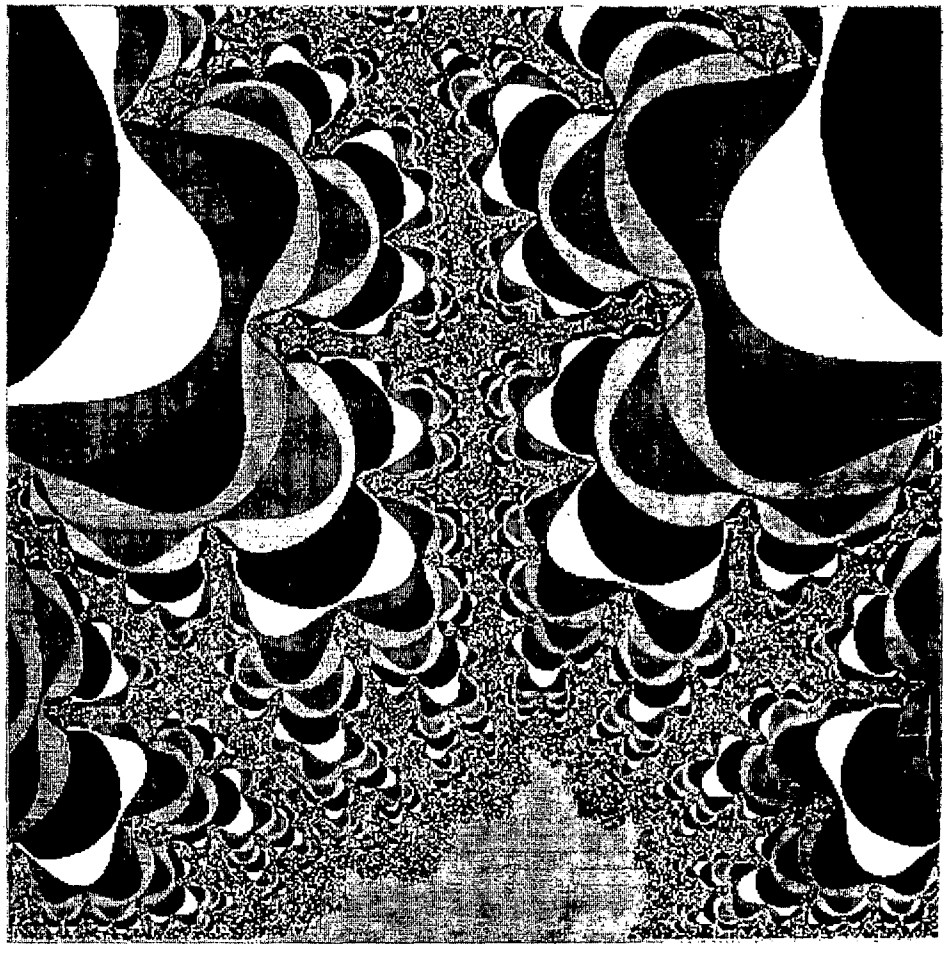
Realna os	Imaginarna os
-0.76 do -0.74	0.01 do -0.03
-1.26 do -1.24	0.01 do -0.03
-0.27 do -0.26	0.00 do -0.01

2. Druga pot je hitrejša, saj ne gledamo obnašanje vsake točke posebej. Najprej poiščemo inverzno enačbo, to pomeni, da funkcija premika točke v obratni smeri. S tem lahko eno točko premikamo ponavljajoče in ta bo skakala okoli fraktalne krivulje. Preko iteracij funkcija lahko porine točko daleč od izhodišča, medtem ko druge omeji na določen prostor. Sliko, ki nastane kot rezultat takega postopka, opremimo z barvami. Mejo med barvami imenujemo fraktalna krivulja. Pri tej poti uporabimo enačbo f(z) = C*z*(1-z).

   Posebej zanimiva so sledeča področja kompleksnega stevila C: (3,0), (0,1), (1,0), (1.475,0.906) in (0,1.06).

```
/*************************************************
*                                                *
*    MANDELBROT SET RUNNING ON TRIDENT XEN-16    *
*                                                *
*       Avtor:Andej Tercelj                      *
*                                                *
/*************************************************/

# include <math.h>
# include <stdio.h>
# define X 640          /* velikost zaslona v x smeri */
# define Y 480          /* velikost zaslona v y smeri */
# define MAX 450        /* velikost okna */
# define OFFSET 0       /*prostor za napis*/
# define A (X/2)-(MAX/2)
# define B (Y/2)-(MAX/2)+OFFSET

main()
{
        int x,y,iter;
        float angleR , angleI ;
        float stran , vel;
        double cr,ci,zr,zi,a,b,l;

        puts("V kateri tocki zelis koordinatno izhodisce? <Re,Im>\n");
        scanf("%f%f",&angleR,&angleI);
        puts("Kaksno velikost stranice zelis? \n");
        scanf("%f",&stran );
        vel = stran/MAX;

        grinit();
        setcol(5);
        rect(A-1,B-1,MAX+2,MAX+2,2);    /* narise okvir v rumeni barvi */
        setxy(A,B);                     /* postavim se v izhodisce */

        for( y=0 ;y<MAX ;y++) {
                for ( x=0 ;x<MAX ;x++){
                        cr=x*vel+angleR;
                        ci=y*vel+angleI;
                        zr=cr;
                        zi=ci;
                        iter=0;
                        do {
                                a=zr*zr;
                                b=zi*zi;
                                l=sqrt(a+b) ;
                                zi=2*zr*zi+ci;
                                zr=a-b+cr;
                                if (l >= 2. ) break;
                        } while( iter++<100);
                        setcol(iter%8); /* postavi eno od osmih barv */
                        dot(A+x,B+y);   /* narise tocko  v x,y */
                }/*  X  */
                 /*  Y  */

        puts ("Zelis sliko printat <y/n>: ");
        if(toupper (getchar()) != 'N' ){
                chdcpy (0,0,639,479,8,1);/* barvni hardcopy on Fujitsu DPL24C */
                grend();}
```

Slika 8. Izpis programa ki oblikuje slike 6 in 7



slika 7. Povečan del grafične slike 6

V. ZAKLJUČEK

Fraktali imajo neštetо obrazov. V sebi skrivajo od simetričnih oblik do zmajevskih potez, štiridimenzionalnih posasti do lepih organskih obrisov. Vsem pa je skupna lastnost rekurzija, ki je numerično in časovno zahtevna za vsak računalnik. S pomočjo pete generacije novih računalnikov bodo procesi hitrejši in uporabnejši v praksi. Današnja generacija slike 7 na hišnem računalniku (tipa Commodore 64) traja dva dni, oziroma dve uri na Deltinem mikroračunalniku Triglav s procesno glavo iAPx286. Danes se področje uporabe naglo širi od filmskih platen do industrije (tekstil, tapete ipd.) pa vse do drugih, predvsem barvnih grafičnih izdelkov. Tako ima fraktalna geometrija z novo generacijo računalnikov neomejene perspektive v interpretaciji kvalitetno-kvantitativnih struktur.

VI. LITERATURA IN NADALJNJA BRANJA:

1. A.K. Dewdney : Computer recreation: A computer microscope zooms in for a look at the most complex object in mathematics. Scientific American, August 1985 , str.8-14.

2. D. Malmberg : Fractals and other diabolical designs, Commodore powerplay , Juni/Julij 1986, str.88-92.

3. A.K. Dewdney : Computer recreation: Wallpaper for the mind;computer images that are almost, but not quite, repetetive. Scientific American, September 1986 , str.14-24.

4. P.R. Sorensen : Fractals. Byte , September 1984 ,str.157-172.

5. A.K. Dewdney : Computer recreation: Of fractal mountains, graftal plants and other computer graphics at Pixar. Scientific American, December 1986 , str.14-20.

6. McDermott : Geometrical Forms Known as fractals find sense in chaos. Smithsonian , December 1983.

7. B.B. Mandelbrot : The fractal geometry of nature. W.H.Freeman, 1982.

RESEARCH OF COMPUTERS AND INFORMATION FOR THE NEXT DECADE

INFORMATICA 3/87

UDK 681.3.001.1:519.6

Anton P. Železnikar
Iskra Delta, Ljubljana

Abstract. This paper offers an open platform for academic discussion on the possibilities of research in the field of future computer architecture and information. It covers the following topics: an introductory survey of system parallelness and intelligence as main criteria for future orientations, how to develop a philosophy of information and conjoin it with other scientific disciplines, the problems of informational machine implementation and of informational programming, hints about new technological possibilities, differences concerning living and artificial systems, and a conclusion relating the strategy of computer industry investment, research, technological design, and marketing for the next decade.

1. Introduction

Evidently, the understanding of computers and their application is developing more and more in the direction of informational machines and informational programs. From this basis, information is becoming the substance of the so-called future intelligent systems. It is upon these new concepts, which include new arising notions, comprehension and possible implementations, that future parallel and multiprocessor computer systems and their intelligent properties are being founded. These objectives are already built-in as basic motives in all of today's academic, governmental, and industrial projects of new generation computer methodology and technology. Projects with such orientation are found in Japan, USA, Great Britain, Germany, France, Soviet Union, India, China, etc. and have also begun to develop in Slovenia (under protection of universities and computer industry).

Researching, technological, and developmental foundations in the field of new computer systems are based on massively parallel and non-massively parallel multiprocessor supercomputers and computer architectures, on newly arising methodologies of artificial, machine, or programming intelligence, and particularly on new design philosophy of understanding, conceptions, architectures, methodologies, and technologies of computer systems (1, 3). In this context, the insufficiency of intelligent machines and intelligent programs on the basis of rationalistic-traditional research and development is being clearly recognized.

2. Philosophy of Information

Intelligence of machines and intelligence of programs can be implemented merely by considering a consequently and basically informational comprehension, which is grounded in several informational principles, i. e., Informing, embedding, arising, and counter-Informing of information and in several properties of these principles, i. e., circularity, parallelness, serialness, and recurrence of information. In information philosophy, Informing means processing of information (Informing, embedding, arising, counter-Informing) and circularity means the regular dynamic property of information (recurrence, parallelness, serialness). In the realm of this comprehension, information is informing circularly (recurrently, parallel, and serially) by the coming of information into existence, by the embedding of information in existing information, and by the coming of counter-information into existence. Through this comprehension, information is performing as a system, which is embedded in information itself.

Informing in the broadest sense (as Informing, embedding, arising, and counter-Informing with properties of circularity as recurrence, parallelness, and serialness) is the substance, essence, and technological imperative of intelligence, i. e., the necessary condition of intelligent reasoning. Probably, in the next decade, a new philosophy of information will successfully develop (3) to such a stage that on this basis a stepwise construction of a fundamental and formally soft information theory will became realistic and developmentally possible. Through this theory, real intelligent concepts and a series of other higher informational functions will be developed, which will be researched in both, what is technologically possible and in living informational mechanisms.

Already, in the current direction of research and development, intelligence is only a particular informational phenomenon (an informational category) which structures and organizes other higher and lower informational functions in different ways. This functions are, for example, motivation, association, strategy, ontological world model, specific organizational forms, and several typical cortical and technological information

processes. In this way, sooner or later, the notion of intelligence will become a senseful complexity of more fundamental, but also higher informational processes. At this stage, intelligence is being understood as a highly structured, organized, and connected information, as well as a complex and mutually interwoven informational arising and processing.

Philosophy of information, theory of information, and the derived understanding of informational machines and programs from these disciplines will be characteristically interdisciplinary oriented research activities. They will mutually refer, unite, and articulate several scientific disciplines, e. g., neural sciences (2) (neurophysiology, neuropharmacology, neuropsychology, neurolinguistics, neuroatonomy, genetics, biology, biochemistry, science of medicine, etc.), understanding of behavior and learning (experimental psychology, psychiatry, neurology), informatics (understanding, cognition, and understanding of cognition of living and technological machines and programs), and information technology (new architectures, methodologies, informational. technological circuits).

3. Informational Machine

A technological machine is in no way a copy of a living machine or organism. An automobile is not like a horse, an airplane is not like a bird, a computer is not a copy of the brain, and a robot is not like a living being. In spite of this, the designing of machines is becoming more and more connected with philosophy and knowledge concerning life, e. g., with the philosophy of Being, experience, and information and with the knowledge concerning brain structure and organization, neurophysiological processes, and functions of thought. To the majority of professionally molded scientists and engineers it seems quite incredible that philosophical thinking can have a practical value for their work (1, 3). Hard sciences and engineering which have their roots in European rationalistic tradition are becoming invalid, helpless, inefficient, and inadequate for solving modern informational problems. This deficiency is coming to the surface in a most distinct way, as mode of thinking, and as methodology of artificial intelligence and of research which concerns the development of new-generation computers, for which project designers have promised and forecasted intelligent behavior.

Knowledge and philosophy of thinking, which concern the nature of biological existence, language, and of human behavior are essential for design, construction, and production of machines and their programs. The objective of today's strategists, designers, planners, builders, and engineers is intelligent machines and intelligent programs. It is becoming evident that intelligence as a technological property cannot be reached without fundamental and comprehensively new research of intelligence's informational background.

From an informational perspective, an informational machine is becoming the synonym for an intelligent machine. On the way to machine intelligence, methodology and technology must assure informational arising in the subsequent technological substance (in material architecture of a new computer) and in information belonging to this substance (by variable, methodologically non-structured programs). Arising which must be implemented in computer architecture is a dynamically, signal-dependently, message-variably, or generally informationally controlled (switched) basic architecture. This is the so-called dynamic machine architecture. The effect of this control dynamics is a virtual architectural variability with actual, virtual, and needed dynamic consequences. Architectural variability is modular on the elementary level, subsystem-structured, and finally system-structured. In this respect, it is variable in its details (in the finest technological structure) and in its subsystems (in a rough machine-linguistical structure).

New computer systems will have to satisfy some given standarization requirements concerning their parallel and multiprocessing architectures. These standards will arise on the basis of experimental, application-dependent, industrial, and marketing experience which will be a consequence of marketing successfulness, of readiness-to-hand (administration, business, and special purpose parallel computer and supercomputer systems), and of process-technological applicability (process-applicable or industrial parallel computer systems).

4. Informational Programming

An informational machine using dynamic architecture will enable the execution of the so-called informational programs. An informational program will be a program which is characteristically non-structured, variable during its runtime, and arising. Today's computer programs do not have these properties or they have them only in a characteristically trivial form. Even now, an informational program, which would be intelligent, would remain merely an objectively non-reachable result of today's programming methodology. Currently, programming methodology is framed by programming-developmental structuring rules which limit the function and disable the arising of programs on the level of today's programming tools, e. g., compilers and programming generators for higher programming languages.

The development of a massively parallel, non-massively multiprocessing, and new-generation computer systems is being confronted with the so-called crisis of programming. After a new machine or new-generation computer is architecturally implemented, the first two problems which arise are its operating system implementation (determination and elaboration of a system, which is an applicable functional collection of mutually compatible basic programs for a general or dedicated machine usage) and its compatibility with machines of a previous technological generation. A development and implementation of these programs, by considering operational and compatibility criteria, could be demonstrated as practically insufficient or even impossible. In this way, some large developmental projects are not reaching objectives which were stated at their beginning. Goals which in general cannot be reached are, for example, intelligent system behavior, intelligent man-machine speech communication, expert properties of computer systems (comparable to human experts), communication in written natural language, etc. Simultaneously, the programming crisis is a crisis of artificial intelligence and of its rationalistic methodology (of matematization, algorithmization, formal linguistics, scientific hardness).

An essential element of the programming crisis is the complexity of operational and applicative programs. This element of crisis is

practically solvable by the development of programming tools and of course, by a sensitive project management of large programming teams.

In the framework of new research and development projects a phenomenon is arising which could be called the project crisis. From the beginning, new projects in the field of informational machines, informational programming, and also informational technology need to be philosophically scrutinized with the point of view of what is technologically and methodologically possible and not so much from the point of view based on the persuasion of performance announcements which do not consider real possibilities of implementation.

5. Informational Technology

Today's informational technology is based on electronic integrated circuits, optical data tranceivers, electromagnetical, optical, and electromechanical storage and peripheral devices, telecommunication equipments, etc. Still, in integrated circuits, semiconductor and conductor elements for normal and low temperatures (supraconductivity) are predominant. However, future development will also have to consider applications of other electromagnetical, antenna, wave-guide, biological, quantum mechanical, neural circuits, etc. which will be technologically and biologically hybrid and integrated in the form of elementary functional units.

The technological foundation of informational machines will be innovative in several aspects. The application of new elements and processes of solid state physics, biochemistry, genetics, and other biological substances will enable the necessary functional diversity and informational arising in physical, logical, and application-linguistical machine structure. In this manner, with a new substantial and processing structure, the information machine will approach in the direction of parallel-processing diversity, which will become functionally comparable to a living neuron substance.

6. Living and Artificial Systems

From molecules of life to cortices, the central nervous systems remains an exemplary model of a senseful informational system, which we wish to copy by technological machines. At this point, a new interdisciplinary understanding of neural science and information science as well as some other marginal fields (philosophy, psychology, technology) is coming into the foreground. As a discipline, artificial intelligence is on the way to reconstruct its own foundation and to recognize in itself a new orientation (1). The research of living and artificial systems is becoming more and more interdisciplinary dependent and accelerated. In this respect, the organization of interdisciplinary research projects and teams in academia and industry are becoming urgent; otherwise it will be not possible to expect the necessary results of research which would guarantee a sufficient progress in the field of intelligent machines and intelligent programs.

7. Conclusion

Research and development of computers and information which are based on a new, informational, and intelligent foundation will be more intensively pursued in the next technological era. Today's parallel, supecomputational, and vector systems, which in connection with today's artificial intelligence and through their functional performances articulate the so-called fifth computer generation, will not perform intelligently. However, on this basis, some standards concerning parallel architectures, system buses, operative systems, communication and graphics standards and new fields of application will be put into effect. Through this development, new technologically advanced, academic, and industrial projects will come into existence.

Even stronger than before, the computer industry will stay on the frontiers of science as the main carrier of research and development in the field of intelligent machines and intelligent methodologies. As up to now, and in the future, academic research will remain limited to preliminary, marginal, and primarily initializing projects. Sooner or later, the computer industry has to take into account its larger investment into the development of new system concepts, designing and programming methodologies, parallel architectures, parallel operating systems, system intelligence, intelligent communication, new application areas, etc. In this new situation of development, the so-called national computer industries cannot expect essential advantages, benefits, or privileges on national markets or a considerable social support. Only the development of international import, export, exchange of marketing knowledge, and business success can yield a real factor of progress for the computer industry.

References

(1) T. Winograd and F. Flores: Understanding Computers and Cognition: A New Foundation for Design. Ablex Publ Corp, Norwood, NJ (1986).

(2) E. R. Kandel and J. H. Schwartz: Principles of Neural Science (Second Edition). Elsevier, New York (1985).

(3) A. P. Zeleznikar: On the Way to Information. Informatica 11 (1986), No. 1, pp. 4-18.

(4) A. P. Zeleznikar: Information Determinations I. Informatica 11 (1986), No. 2, pp. 3-17.

PREGLED ARKITEKTUR TER UPORABLJENIH ISO STANDARDOV V MAP/TOP LOKALNIH MREŽAH

INFORMATICA 3/87

UDK 681.324.087

Jože Rugelj, Marijan Vidmar
Institut »Jožef Stefan«, Ljubljana

Članek opisuje razvoj in trenutno stanje MAP-a in TOP-a. MAP (Manufacturing Automation Protocol) definira množico standardov za komunikacije v avtomatizirani računalniško vodeni proizvodnji, TOP (Technical Office Protocols) pa komunikacijske standarde za poslovno in inženirsko uporabo.

This paper describes the development and the present situation of MAP and TOP. MAP (Manufacturing Automation Protocol) is a selection of currently available standards for manufacturing automation. TOP (Technical and Office Protocols) specification is intended to provide data communications for the technical and office environment.

1. UVOD

Boj za konkurenčnost sili proizvajalce v višjo stopnjo avtomatizacije. Danes avtomatizacija posameznih procesov v proizvodnji ne zadošča več. Za bolj optimalno vodenje je potrebno dinamično spremljati in upravljati celoten proizvodni proces. Računalniško podprto snovanje (CAD) in proizvodnja z računalniško krmiljenimi stroji ter roboti (CAM) tvorijo integriran, računalniško podprt proizvodni sistem (CIM). V CIM se funkcije načrtovanja, proizvodnje, vodenja in upravljanja prepletajo in omogočajo boljši pretok informacij znotraj sistema.

Druga prednost, ki jo daje tak integriran sistem, je izboljšanje kvalitete izdelkov in znižanje proizvodnih stroškov, zelo pogosto pa tudi humanizacija dela. Integrirani sistemi so praviloma tudi zelo fleksibilni. To omogoča hitre spremembe proizvodnih programov in prilagajanje zahtevam tržišča ter zelo hitro uvajanje najnovejših dosežkov v proizvodnjo, saj sta CAD in CAM tesno povezana.

Integrirana proizvodnja zahteva prenos, obdelavo in shranjevanje velike količine podatkov. Pri tem se pojavi problem povezovanja posameznih komponent sistema v enovito celoto. Število proizvajalcev programske in materialne opreme je znotraj sistema običajno veliko, oprema različnih proizvajalcev pa praviloma ni direktno združljiva in to onemogoča komunikacijo. MAP (Manufacturing Automation Protocol) in TOP (Technical and Office Protocols) predstavljata poskus rešitve tega problema.

2. ZGODOVINSKI RAZVOJ

Ameriška avtomobilska industrija ima velike probleme z japonskimi konkurenti in zaradi tega so se lotili posodabljanja svoje proizvodnje. General Motors (GM) je naletel na velike težave pri povezovanju obstoječe opreme v integriran sistem in zato so se lotili standardizacije. Za njihova prizadevanja na tem področju so se v letu 1984 začeli zanimati tudi drugi uporabniki in proizvajalci računalniške opreme in obdelovalnih strojev ter robotov. S sodelovanjem vseh zainteresiranih pod vodstvom GM so bile določene začetne specifikacije za MAP.

Vzporedno s tem pa se je pojavila tudi potreba po standardizaciji računalniških komunikacij za tehnično in poslovno rabo. Skupino standardov s tega področja so imenovali TOP (Technical and Office Protocols), glavni koordinator pa je bila firma Boeing.

TOP ima iste cilje kot MAP, le da gre pri TOP za drugačno okolje. TOP obravnava komunikacije, ki jih zahtevajo elektronska pošta, procesiranje besedil, izmenjava tekstovnih in drugih dokumentov, prenos datotek, prenos grafike, upravljanje baz podatkov, videotext in poslovne obdelave. Predvidene so priključitve različnih tipov računalnikov, od mikroračunalniških delovnih postaj do velikih računalnikov (mainframes).

V letu 1985 sta se skupini tudi formalno povezali v MAP/TOP User's Group znotraj združenja Society of Manufacturing Engineers (SME). Pomen, ki ga pripisujejo prizadevanjem na področju standardizacije komunikacijskih protokolov za CIM, je mogoče razbrati tudi iz seznama članic MAP/TOP User's Group, če naštejemo samo najpomembnejše: IBM, Hewlett-Packard, DEC, Honeywell, Motorola, Intel, AT&T, General Motors in Boeing.

3. MAP/TOP ARHITEKTURE

MAP in TOP sta specifikaciji za skupino standardov, zasnovanih na ISO/OSI referenčnem modelu, ki omogočajo povezovanje različnih tipov računalnikov in računalniško krmiljenih strojev. Standardi obsegajo vse aspekte komunikacijskega procesa, od fizičnih povezav do visoko-nivojskih aplikacijskih jezikov.

Razlike med MAP in TOP so prikazane na sliki.

LAYERS	TOP V1.0 PROTOCOLS	MAP V2.1 PROTOCOLS
Layer 7 Application	ISO FTAM (DP) 8571 File Transfer Protocol	ISO FTAM (DP) 8571 File Transfer Protocol, Manufacturing Messaging Format Standard (MMFS) and Common Application Service Elements (CASE)
Layer 6 Presentation	NULL (ASCII and Binary Encoding)	
Layer 5 Session	ISO Session (IS) 8327 Basic Combined Subset and Session Kernel, Full Duplex	
Layer 4 Transport	ISO Transport (IS) 8073 Class 4	
Layer 3 Network	ISO Internet (DIS) 8473 Connectionless and for X.25 - Subnetwork Dependent Convergence Protocol (SNDCP)	
Layer 2 Data Link	ISO Logical Link Control (DIS) 8802/2 . (IEEE 802.2) Type 1, Class 1	
Layer 1* Physical	ISO CSMA/CD (DIS) 8802/3 (IEEE 802.3) CSMA/CD Media Access Control, 10Base5	ISO Token Passing Bus (DIS) 8802/4 (IEEE 802.4) Token Passing Bus Media Access Control

MAP predvideva dve osnovni arhitekturi:

- MAP hrbtenična arhitektura (backbone architecture),
- MAP celična arhitektura (cell architecture).

Trenutna specifikacija MAP (verzija 2.1) vključuje samo prvi tip arhitekture, vendar sta v kasnejših verzijah (verzija 2.2) predvidena oba tipa. V hrbteniční arhitekturi imajo vse priključene postaje implementiranih 6 od 7 ISO/OSI nivojev. V trenutni implementaciji še ni definiran predstavitveni (6.) nivo. Vsaka postaja lahko komunicira z vsako drugo postajo v sistemu ali tudi izven njega (preko javnega digitalnega komunikacijskega omrežja). Standard določa uporabo širokopasovnega vodila (broadband bus).

Za aplikacije, kjer je zahtevan prenos časovno kritičen, so predvidene podmreže s celično arhitekturo. Postaje, priključene na take podmreže, naj bi imele implementirana le prva dva in sedmi nivo po ISO/OSI modelu. Predvidena je uporaba enopasovnega vodila z moduliranim nosilnim signalom (carrier - band bus) brez frekvenčnega multipleksiranja. Taka oblika je izbrana zato, ker je v primerjavi z osnovnim pasom (baseband) mnogo manj občutljiva za motnje, ki so tipične v industrijskem okolju. V podmreži pa bi bila tudi vmesniška postaja, ki bi omogočala priključitev podmreže v hrbtenično arhitekturo.

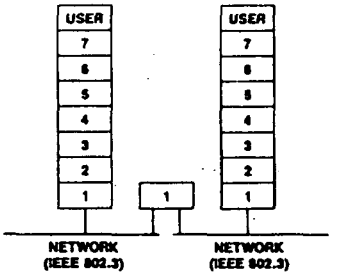
Tako arhitekturo imenujejo tudi EPA (Enhanced *Performance Architecture*).

TOP predvideva arhitekturo, ki je identična MAP hrbteniční arhitekturi.

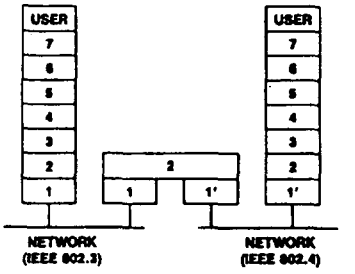
MAP/TOP predvideva povezovanje mrež ali mrežnih segmentov, ki so lahko tudi geografsko zelo oddaljeni.

Za povezovanje uporabljamo 4 arhitekturne elemente: repeater, bridge, router in gateway.

Repeater je naprava, ki je namenjena povezovanju mrež na fizičnem nivoju. To pomeni, da morata imeti mreži, ki ju povezujemo, enak fizični nivo. Glavni namen uporabe repeaterjev je povečati dovoljene razdalje med postajami, ki so omejene s fizikalnimi lastnostmi prenosnega medija.

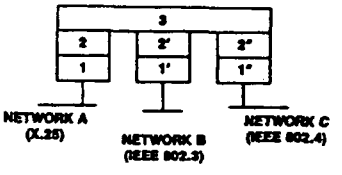


Bridge je namenjen za povezovanje mrež, ki se razlikujejo na 1. in 2. nivoju (npr. CSMA/CD in podajanje žetona).



Obe mreži morata imeti enoten naslovni prostor. Bridge lahko poveča mrežo preko dovoljenih omejitev znotraj segmenta ali fizično loči 2 segmenta v mreži.

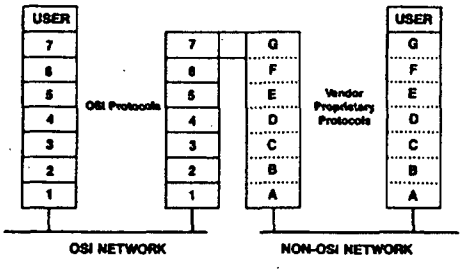
Router omogoča povezovanje na mrežnem nivoju ne glede na tip prenosnega medija in načina dostopa do medija.



Router ima naslov, ki je poznan v vseh mrežah, ki so priključene nanj. Router opravlja izbiro poti in preklaplja sporočila glede na mrežne naslove in stanje priključenih mrež.

Gateway pa omogoča priključitev mrež, ki nimajo OSI strukture. Tu so mišljene predvsem obstoječe

lokalne mreže, javna komunikacijska omrežja in podmreže z nepopolno OSI strukturo.



4. STANDARDI V MAP/TOP

MAP ostaja zvest mrežnim protokolom, ki jih je definirala mednarodna standardizacijska organizacija (ISO). OSI (Open System Interconnection) referenčni model so sprejele tudi vse druge standardizacijske organizacije. Del MAP-a je zasnovan na priporočilih in standardih IEEE 802, pri čemer nekateri še niso dokončno definirani (draft standards, draft proposal).

Podnivo kontrole dostopa do medija (MAC) je v novejših verzijah MAP/TOP postavljen na fizični nivo, ker taka razporeditev bolje ustreza konceptu tega podnivoja. Tudi ISO predvideva tako spremembo IEEE definicije MAC podnivoja.

Fizični nivo OSI referenčnega modela omogoča fizično povezavo za prenos podatkov med postajami v mreži. Izvaja pretvorbo podatkov v električne signale za prenos po mediju in upravlja dostop do posameznega medija.

MAP in TOP se razlikujeta pri izbiri standardov za fizični nivo.

V TOP je kot osnovni medij izbran 50 ohmski koaksialni kabel, ki omogoča prenose do 10 Mbps na 500 m dolgih mrežnih segmentih in pri maksimalnem številu 1023 postaj. Tak kabel je izbran zaradi velike razširjenosti tega medija zaradi uporabe Ethernet-a, zaradi enostavnega prehoda iz Ethernet-a na IEEE 802.3 CSMA/CD ter možnosti hkratne uporabe obeh protokolov in zaradi zanesljivosti in dobrih izkušenj pri uporabi tega medija v preteklosti. CSMA/CD IEEE 802.3 je izbrana metoda dostopa do medija, ker je poleg žetona na obroču zaradi svojih značilnosti najbolj primerna za TOP okolje. Specifikacije medija in protokola za CSMA/CD MAC so podane v /ISO 8802/3/ ali v /IEEE 8023/.

TOP predvideva v bodoče še vključitev drugih medijev in MAC protokolov, ki bi bolj ustrezali specifičnim potrebam uporabnikov. Predvidena je uporaba IEEE 802.5 Token Passing Ring na oklopljenih paricah in IEEE 802.6 Metropolitan Aera Network na optičnih vlaknih.

V MAP je za osnovni medij izbran 75-ohmski (CATV) koaksialni kabel za hitrosti do 10 Mbps. Širokopasovno vodilo (broadband) ima zelo veliko kapaciteto prenosa podatkov. Poleg tega omogoča hkratno prenašanje govora in slike (video). Razdalje ne predstavljajo problemov, saj so signali modulirani in se ojačujejo v glavi na koncu kabla (head-end), kjer se izvrši tudi frekvenčni premik signala. Vse to omogoča veliko imunost proti motnjam, ki so zelo pogoste v industrijskem okolju. Zaradi velike frekvenčne širine je možno sočasno delovanje do 100 lokalnih mrež na istem kablu, kar je zelo koristno pri uvajanju MAP-a v delujoče tovarne, kjer ni mogoče vseh naprav povezati v enoten komunikacijski sistem v enem koraku. Poleg tega je CATV kabel že instaliran v mnogih tovarnah. Žeton na vodilu (token passing bus) je izbrana metoda dostopa do medija. Ta metoda je primerna za MAP, saj zagotavlja vsaki postaji dostop do medija v nekem vnaprej določenem času in zagotavlja konstanten pretok podatkov tudi pri velikih obremenitvah.

Linijski nivo OSI referenčnega modela vzpostavlja in upravlja komunikacijsko povezavo med dvema postajama. Funkcije tega nivoja so še kreiranje paketov, preverjanje pravilnosti prenosa, naslavljanje in ostale funkcije potrebne za točen prenos podatkov med postajami. Delovanje linijskega nivoja je neodvisno od izbrane metode dosega uporabljene na fizičnem nivoju, in obratno.

IEEE 802.2 specifikacija za logično kontrolo zveze (LLC) je ISO standard za linijski nivo /ISO 8802/2/. Standard določa tri tipe sistemov:

1. brezpovezavno orientiran tip,
2. povezavno orientiran tip in
3. tip s posameznimi podatkovnimi paketi (single frame).

Tretji tip je predviden za podporo aplikacij v realnem času v MAP EPA arhitekturi. V standardu sta dva razreda servisov: 1. razred, brezpovezavni servis, ki vsebuje samo sisteme tipa 1 in 2. razred, povezavno orientiran servis, ki vsebuje vse tri tipe.

Sicer je za MAP in TOP izbran sistem tipa 1, servis razreda 1. Ta omogoča zvezo med dvema postajama brez vzpostavljanja povezave. Ta tip ne omogoča oštevilčenja paketov, potrjevanja kontrole pretoka in popravljanja napak. Za vse to poskrbijo višji nivoji.

Mrežni nivo skrbi za preklapljanje in usmerjanje paketov med postajami v isti mreži ali med več mrežami, neodvisno od uporabljenega transportnega protokola. Mrežni nivo vrši tudi kontrolo pretoka. Servisi mrežnega nivoja so neodvisni od razdalj med mrežami.

Funkcije nivoja 3 v MAP in TOP so enake, le da TOP uporablja novejši arhitekturni opis, kjer je nivo razdeljen na štiri podnivoje.

Podnivoji so povezani s spreminjanjem informacije o globalnem naslovu v informacijo o usmerjanju paketa v mrežah, vzdrževanju tabel ali algoritmov za usmerjanje, vzpostavljanjem ter prekinjanjem povezav in preklapljanjem vhodnih sporočil v vozlišču na pravo izhodno pot. Del informacij zagotavlja tudi podnivo aplikativnega nivoja, imenovan upravljanje mreže (network management).

Uporabljeni brezpovezavni mrežni protokol (CLNS) je definiran s standardom ISO DIS 8473 /ISO 8473/.

Transportni nivo (nivo 4) skrbi za zanesljiv prenos podatkov za osebke na nivoju seje, ne da bi se le-ti morali ukvarjati s podrobnostmi za zanesljiv podatkovni prenos.

Transportni nivo tudi optimizira uporabo razpoložljivih mrežnih servisov nivojev 1, 2 in 3, da izpolni zahtevek nivoja seje.

Protokol definiran za transportni nivo, zagotavlja povezavo med končnimi uporabniki, med dvema osebkoma na transportnem nivoju.

Transportne funkcije na transportnem nivoju zagotavljajo zahtevano kvaliteto servisa, ki je odvisna od kvalitete servisa mrežnega nivoja.

Transportni protokol definiran s standardom ISO IS 8073 /ISO 8073/ je povezavno orientiran protokol, ki ga sestavlja pet razredov servisov. Servisi so razvrščeni po zahtevnosti. Najbolj kompleksen je razred 4.

Servis razreda 4 opravlja kontrolo pretoka in omogoča združevanje več transportnih povezav v eno samo. Servis tega razreda tudi dopušča uporabo zanesljive mreže in zna odkrivati in popravljati napake s preverjanjem vrstnega reda sporočil in detekcijo izgube paketov ali njihovo poškodbo. Zaradi teh lastnosti je v MAP/TOP izbran razred 4.

Povezavno orientiran transportni servis je sestavljen iz dveh delov: upravljanja povezave in prenosa podatkov. K upravljanju povezave spadajo vzpostavitev povezave (connect service), prekinitev povezave (disconnect service) in poročanje o statusu povezave (status service).

Prenos podatkov podpirata dva servisa, normalni podatkovni prenos in pospešeni podatkovni prenos (expedited data service), ki omogoča prenos omejene količine podatkov izven normalnih podatkovnih tokov.

Nivo seje zagotavlja organizacijo in sinhronizacijo komunikacije med osebkoma na nivoju predstavitve in upravlja izmenjavo podatkov. Za upravljanje prenosa uporablja transportno povezavo. V času vzpostavljene povezave na nivoju seje servisi nivoja seje služijo osebkom na predstavitvenem nivoju za urejanje konverzacije in zagotavljajo urejeno izmenjavo sporočil na povezavi.

Nivo seje razširja transportni servis z mehanizmi za upravljanje in struktuiranje zanesljivega podatkovnega prenosa, ki ga zagotavlja transportna povezava. Povezava na nivoju seje je lahko strukturirana tako, da se izmenjave podatkov med uporabniki izvajajo simultano dvosmerno (full-duplex), izmenično dvosmerno (half-duplex) ali enosmerno (simplex). ISO protokol nivoja seje predstavljajo 4 podmnožice servisov: jedro, osnovna kombinirana podmnožica, osnovna sinhronizirana podmnožica in podmnožica osnovnih aktivnosti. V jedru so osnovne funkcije: vzpostavitev in prekinitev povezave ter prenos podatkov. Osnovna kombinirana podmnožica poleg funkcij iz jedra omogoča še pospešen prenos podatkov (expedited data) ter upoštevanje izida pogajanja o parametrih povezave in je lahko simultano dvosmerna ali izmenoma dvosmerna (half-duplex).

MAT/TOP specifikacija uporablja ISO Basic Connection - Oriented Session Service Definition (ISO 8326) in ISO Basic Connection - Oriented Session Protocol Specification /ISO 8327/.

Implementacija ISO nivoja seje sestavljata jedro in dvosmerna simultana podmnožica osnovne kombinirane podmnožice.

Predstavitveni nivo v MAP/TOP je v trenutnih verzijah prazen.

Aplikacijski nivo, sedmi, najvišji nivo v OSI referenčnem modelu omogoča uporabniku ali aplikaciji uporabo servisov, ki jih nudijo nižji nivoji. Uporabnik dosega in uporablja mrežne servise preko množice mrežnih orodij (procedurnih knjižnic in mrežnih uslužnostnih programov), ki so realizirani na aplikacijskem nivoju. Na sedmem nivoju MAP določa uporabo petih protokolov.

CASE (Common Application Service Elements), ki je določen s standardom /ISO 8650/, MMS (Manufacturing Message Service) z osnutkom standarda RS-511, upravljanje mreže (Network Management) določenim z /IEEE 8802/1/ in FTAM (File Transfer, Access and Management), ki ga določa /ISO 8571/.

TOP na aplikacijskem nivoju določa le uporabo FTAM.

CASE služi za vzpostavljanje komunikacije, torej za povezavo med MAP aplikacijami. CASE tudi skrbi za upravljanje in prekinitev izmenjave podatkov. Ostali štirje protokoli na aplikacijskem nivoju uporabljajo CASE.

RS-511 je nadzorni jezik, ki je bil razvit za lokalne mreže v industrijskem okolju in omogoča centralnemu računalniku nadzor in vodenje perifernih računalnikov, robotov in numerično-krmiljenih strojev. V EPA arhitekturah pri aplikacijah v realnem času je možna tudi direktna komunikacija z linijskim nivojem.

Servisi RS-511 so razvrščeni v naslednje opisne enote:
Splošni servisi, dostop do spremenljivk v perifernih enotah, spremljanje dogodkov v perifernih enotah, neposredno krmiljenje perifernih naprav, upravljanje in usklajevanje dostopa do kritičnih zmogljivosti, razvrščanje poslov, enostaven prenos in upravljanje z datotekami, komunikacija z operaterjem, beleženje poteka proizvodnega procesa in nalaganje programov na periferne enote.

Splošni servisi omogočajo uporabniku, da poizveduje o definiranih imenih v drugih RS-511 sistemih, izbrišejo neizvršene zahtevke, ki jim je potekel čas in vrnejo sporočilo za sporočilo, ki ga ni mogoče pravilno dekodirati.

Funkcija dostopa do spremenljivk v perifernih enotah omogoča branje, pisanje, definicijo spremenljivk in definicijo tipov. Branje in pisanje ohranjata tip podatka, kar je pomembno zaradi različnosti interne predstavitve podatkov v različnih sistemih. Možno je tudi oblikovanje sestavljenih spremenljivk, kjer je z enim namenom definiranih več enostavnih spremenljivk, kar predstavlja prenašanje logično povezanih podatkov.

Spremljanje dogodkov v perifernih napravah omogoča hitro ukrepanje v določenih primerih in sinhronizacijo ostalih dejavnosti glede na določene dogodke. Neposredno krmiljenje omogoča neposredno krmiljenje določenih naprav v sistemu. Upravljanje in usklajevanje dostopa do kritičnih zmogljivosti je potrebno zaradi nezdružljivosti določenih akcij.

Servis razvrščanja poslov izbira posel, ki ga mora izvršiti enota, ki je končala prejšnji posel. Osnovni protokol v MAP za delo z datotekami je sicer FTAM, vendar je le-ta za določene primere preveč kompleksen. Zato RS-511 določa enostaven nabor servisov za delo z datotekami.

Komunikacija z operaterjem omogoča izpis omejene količine informacije, kot so navodila operaterju pri stroju in sprejem informacij od operaterja. Za pretok večjih količin so definirani servisi ISO VTP (Virtual Terminal Protocols). Beleženje poteka proizvodnje omogoča zapis in shranjevanje pomembnih parametrov proizvodnega procesa, ki je nujno v nekaterih dejavnostih (npr. farmacevtska industrija).

Nalaganje programov v periferne enote se razlikuje od dela z datotekami zato, ker mora biti enota in njene vhodno/izhodne vrednosti v znanem stanju, ko naložimo program. V nasprotnem primeru lahko pride do nepredvidenih dogodkov, ki poškodujejo izdelke ali naprave.

Servis upravljanja mreže zagotavlja, da je proces prenosa podatkov učinkovit, pravočasen, zanesljiv in varen. Upravljanje mreže lahko razdelimo v dve funkcionalni enoti: zbiranje podatkov o mreži in vodenje in nadzor mreže. Oboje je povezano z izkoriščenostjo, statusom, položajem, zmogljivostjo ter razpoložljivostjo in napakami elementov mreže.

Funkcije servisa upravljanja mreže omogočajo postavitev mreže v stanje, v katerem lahko prenaša podatke, reševanje iz napak, rekonfiguracijo mreže in zaključitev dejavnosti v mreži /ISO 8021/.

V MAP verziji 2.1 je uporabljena skromnejša verzija servisa upravljanja mreže, ki je bila zasnovana kot GM MAP Network Management Requirements Specification in bo predvidoma vključena tudi kot podmnožica ISO specifikacije.

Servisi FTAM za prenos, doseg in upravljanje z datotekami so definirani z mednarodnim standardom /ISO 8571/. Logično je razdeljen v dva dela, protokol za prenos datotek in servisi dosega in upravljanja, ki se ukvarjajo z zaščito in atributi datotek. V MAP je možno pri enostavnem delu z datotekami namesto FTAM uporabiti tudi servise RS-511.

V kasnejših verzijah TOP-a bodo definirani še drugi protokoli, kot so elektronska pošta, izmenjava dokumentov, različni tipi serverjev in virtualni terminal.

5. ZAKLJUČEK

S stališča uporabnikov in proizvajalcev opreme za CIM se zdi razvoj MAP/TOP prepočasen. Potrebe po opremi za CIM zelo hitro naraščajo, proizvajalci in uporabniki pa želijo dokončne standarde zanje.

Sprejemanje standardov je v splošnem dolgotrajen postopek, ki vključuje pogajanja med različnimi komiteji in delovnimi skupinami od zdru'enj inženirjev ter združenj proizvajalcev pa do mednarodnih standardizacijskih organizacij.

V primerjavi s podobnimi postopki je bila pot MAP-a skozi vse te postopke hitra zahvaljujoč široki in stalni podpori vseh zainteresiranih z GM na čelu /CONTROL 86/.

6. REFERENCE

/CONTROL 86/ CONTROL ENGINEERING, October 1986

/IEEE 8021/ IEEE Project 802, Local Area Network Standards, IEEE Standard 802.2 Network Management

/IEEE 8022/ IEEE Project 802, Local Area Network Standards, IEEE Standard 802.2 Logical Link Control (ISO DIS 8802/2)

/IEEE 8023/ IEEE Project 802, Local Area Network Standards, IEEE Standard 802.3 CSMA/CD Access Method and Physical Layer Specification (ISO DIS 8802/3)

/IEEE 8024/ IEEE Project 802, Local Area Network Standards, IEEE Standard 802.4 Token Passing Buss Access Method and Physical Layer Specification

/ISO 8072/ Information Processing Systems - Open Systems Interconnection: Transport Service Definition, ISO IS 8072, 1984

/ISO 8073/ Information Processing Systems - Open Systems Interconnection: Transport Protocol Specification, ISO IS 8073, 1984

/ISO 8326/ Information Processing Systems - Open Systems Interconnection: Session Service Definition, ISO IS 8326, 1984

/ISO 8327/ Information Processing Systems - Open Systems Interconnection: Session Protocol Specification, ISO IS 8327, 1984

/ISO 8473/ Information Processing Systems - Data Communications: Protocol for Providing the Connectionless-mode Network Service, ISO DIS 8473

/ISO 8571/1/ Information Processing Systems - Open Systems Interconnection: File Transfer Access and Management, Part 1: General Description, ISO DP 8571/1, TC97/SC6/N1669, February 1984

/ISO 8571/2/ Information Processing Systems - Open Systems Interconnection: File Transfer Access and Management, Part 2: The Virtual Filestore, ISO DP 8571/2, TC97/SC6/N1670, February 1984

/ISO 8571/3/ Information Processing Systems - Open Systems Interconnection: File Transfer Access and Management, Part 3: Service Definition, ISO DP 8571/3, TC97/SC1671, February 1984

/ISO 8571/4/ Information Processing Systems - Open Systems Interconnection: File Transfer Access and Management, Part 4: Protocol Specification, ISO DP 8571/4, TC97/SC6/N1672, February 1984

/ISO 8571/ Second Draft Proposal of (ISO8571/1), (ISO8571/2), (ISO8571/3), and (ISO8571/4)

/ISO 8650/ Information Processing Systems Open System Interconnection: Common Application Service Elements ISO DIS 8650

/ISO 8802/1/ Local Area Networks - Network Management, ISO DIS 8802/1
ISO version of (IEEE 8021)

/ISO 8802/2/ Local Area Networks - Logical Link Control, ISO DIS 8802/2. ISO version of (IEEE8022)

/ISO 8802/3/ Local Area Networks - Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, ISO DIS 8802/3. ISO version of (IEEE8023)

/ISO 8802/4/ Local Area Networks - Network Management, ISO 8802/4
ISO version of (IEEE 8024)

/MAP21/ General Motors's Manufacturing Automation Protocol: A Communications Network for Open Systems Interconnection, version 2.1, General Motors Corporation, Manufacturing Engineering and Development, Advanced Product and Manufacturing Engineering Staff (APMES), APMES A/MD - 39, GM Technical Center, Warren, MI 48090-9040

/TOP10/ Technical and Office Protocols, Specification version 1.0, Boeing, Seattle, November 1985

UDK 681.3.02.06:519.687.4

Peter Kolbezen, Slavko Mavrič in Branko Mihovilović
Institut »Jožef Stefan«, Ljubljana

Povzetek V neprestani tekmi za čim večje računalniške hitrosti se pojavljajo vedno novi alternativni predlogi, ki obetajo boljše razmerje med ceno in zmogljivostjo računalnika. Novejši od predlogov, ki trenutno zmaguje - še posebej v pogledu možnosti realizacije, je RISC arhitektura računalnika (Reduced Instruction Set Computer). Ta se odlikuje po preprostosti materialne opreme in po skladnosti med arhitekturo in prevajalnikom. Uporabljajo se optimirani prevajalniki, ki prevajajo programske jezike navzdol do nivoja instrukcije. Te instrukcije ne vsebujejo mikroinstrukcij velikega adresnega virtualnega prostora in imajo kolikor mogoče kratek ukazni cikel. RISC arhitektura je ena najmlajših računalniških arhitektur, ki med drugim obeta tudi presenetljivo hitrost procesiranja.

Prispevek daje pregled nad splošnimi principi, značilnostmi in problematiko RISC arhitektur in dosežki na obravnavanem področju.

RISC ARCHITECTURES. As the race for higher speed in computers continues, new alternatives are proposed promising better price/perfomance. One of these new architectures that is gaining momentum, is RISC - Reduced Instruction Set Computers. RISC aim for both simplicity in hardware and synergy between architectures and compilers. Optimizing compilers are used to compile programming languages down to instructions that are as unencumbered as microinstructions in a large virtual address space, and to make the instructuion cycle time as fast as possible. RISC architectures are just around the corner and promise spectacular processing speeds.

The paper gives short survey over general RISC principles, characteristics, problems, and achievements on the area of reduced architectures.

1. UVOD

Razvoj v svetu kaže, da se bodo računalniški sistemi v bodoče nagibali k vse boljı poenotenim in manj raznolikim arhitekturam. Več ali manj novi principi načrtanih računalniških arhitektur, ki so se rojevali od sredine 70-ih let do danes, pa slone na dokaj različnih pristopih.

Obravnavani pristop je pravzaprav presenetljiv za današnji čas. Značilen je po vse boljı kompleksnih in zamotanih računalniških arhitekturah. Večje računalniške zmogljivosti so zahtevale tudi večjo ceno tako za materialno kot programsko opremo. Zgled predstavlja Intelov iAPX 432, ki je bil posebej načrtovan za to, da podpira jezik Ada. V takšnem, ti. CISC sistemu (Complex Instruction Set Computer), je mogoče povečevati računalniško zmogljivost predvsem z dodajanjem posebnih namenskih procesorjev oziroma naprav, kot so procesor za aritmetiko v plavajoči vejici, matrični ali vektorski procesor, posebna enota za upravljanje s pomnilnikom, cache pomnilnik in večkratna vodila. Zaradi takšnih, dokaj neugodnih pristopov, in ne nazadnje zaradi kompleksnih mikroprogramskih arhitektur v CISC istemih z večnivojskimi operacijami, na kakršne naletimo pri VAXu, obstajajo poskusi poenostavljanja procesorskih struktur in rabe takšne programske opreme, ki omogoča čim optimalnejšo računalniško zmogljivost. To je tudi eden od bistvenih razlogov za obravnavani pristop, ki pa niti ni tako nov. Seymore Cray iz Cray Research je bil vodilni zagovornik RISC arhitekture v poslednjih 20-ih letih: arhitekture, ki je bila vse do danes le objekt raziskav velikih raziskovalnih laboratorijev. Najvidnejši raziskovalec na tem področju je David Patterson iz University of California at Berkeley, ki je novi arhitekturi tudi prvi nadel ime RISC.

Z omenjenim premikom v arhitekturi je moč doseči zavidljive hitrosti procesiranja tudi pri manjših strojih, ki se bodo lahko kosali z

DECovim Micro VAXom II. Napori razvijalcev v naslednjih letih bodo še nadalje usmerjeni predvsem v povečevanje procesne moči, ki jo je, kot domnevajo, moč povečati še posebno pri cenenih delovnih postajah vsaj za dva velikostna razreda.

Videti je, da je v pogledu zahtev po naraščanju procesne moči vodilna računalniška arhitektura Risc. Ta arhitektura bo v bodoče, sloneč na ideji poenostavitve osnovnih procesorskih arhitektur, imela dalekosežen vpliv tudi na programsko opremo. Izvedenci na tem področju pričakujejo:

- da bodo delovne postaje že v bližnji prihodnosti delale v območju nad dva do osem MIPSov (miljon instrukcij na sekundo)
- da bodo srednje veliki procesorji (miniračunalniki) delali s hitrostjo 20 do 30 MIPS. Predvidoma bodo takšne hitrosti dosegljive že do konca desetletja
- da bodo procesorji z RISC arhitekturo in zmogljivostjo 2 do 10 MIPS dobavljivi pri nekaterih glavnih prodajalcih že v roku nekaj mesecev do enega leta
- da bodo namenski procesorji za visokonivojske jezike (Lisp, Prolog in druge) zasnovani na RISC arhitekturi
- da se bo razvojni cikel mini in mikroračunalnikov, segajoč od njihove zasnove do proizvoda, zmanišal na leto dni ali celo manj.

2. KARAKTERISTIKE IN PRINCIPI RISC

RISC arhitektura ima nekaj splošnih značilnosti, ki jih je mogoče strniti v naslednje alinneje:

- Manjše število instrukcij, ki se odlikujejo po preprostosti. Značilno število je pod 50 instrukcij, medtem ko ima VAX kar 300 instrukcij.
- Večina instrukcij se izvaja v enem samem ciklu. Preproste so tudi instrukcije tipa load/store.
- Instrukcije so fiksne dolžine. Zato se z materialno opremo lažje dekodirajo.
- Arhitektura je bogata na registrskih skladih, ki upravljajo čip.
- Prevajalnik je integralni del arhitekture. Uporaba programske opreme (prevajalnikov) je primernejša, kot mikrokoda v materialni opremi.
- Arhitektura je bogata na non-von Neumannovih konstruktih in podpira viskonivojske jezike. Tako zmanišuje semantične vrzeli med procesorjem in jezikom.
- Odlično je razmerje v pogledu trženja materialne/programske opreme (pri izkoriščanju VLSI tehnologije).

Prednost RISC arhitekture je v tem,

- da jo je sorazmeroma lahko implementirati v VLSI tehnologiji
- da omogoča gradnjo sistemov, ki so sestavljeni iz posameznih gradnikov (modularnost)
- da omogoča učinkovito prevajanje
- da je mogoče doseči izjemno kratek čas izvajanja programov

RISC arhitektura kaže vse večjo uporabnost tudi v večnamenskem procesnem okolju; trditev, ki je bila doslej močno vprašljiva.

Uspešnost arhitekture sloni na njeni skladnosti s prevajalnikom. Zato, da se takšna skladnost doseže, se uporabljajo naslednji principi:

Regularnost. Nekaj, kar se že izvaja nekje na nek način, se na enak način ne sme izvajati nikjer drugje. Npr., včasih se nastavljanje cc bitov razlikuje po zlogovnih ali besednih operacijah.

	RISC I	68000	Z8002	VAX-11/780	PDP-11/70
Leto proizvodnje	1982	1980	1979	1978	1975
Osnovne instrukcije	31	61	110	248	65
Splošni registri	32	15	14	13	6
Način naslavljanja	2	14	12	18	12
Dolžina adrese	32	24	16	32	16
Frekvenca osn. ure /MHz/	7.5	10	6	5	7.5
Reg.-reg. več./usek/	0.4	0.4	0.7	0.4	0.5
Modif. indek. razveji. če "0" (razvejitev)	1.2	1.0	2.2	1.4	0.8

Slika 1. Primerjava osnovnih značilnosti procesorja RISC1 in nekaterih drugih procesorjev oz. miniračunalnikov.

	IBM 801	RISC1	MIPS
Leto proizvodnje	1980	1982	1983
Število instrukcij	120	39	55
Velikost kontr. pomnilnika	0	0	0
Dolžina instruk. /biti/	32	32	32
Uporabljena tehnologija	ECL MSI	NMOS VLSI	NMOS VLSI
Model izvajanja	reg-reg	reg-reg	reg-reg

Slika 2. Primerjava nekaterih osn.karakteristik RISC procesorjev,ki so bili proizvedeni v letih 1980 do 83.

Ortogonalnost. Podatkovni tipi, naslavljanje in instrukcije morajo biti definirani vsak posebej, v medsebojni neodvisnosti. Npr., včasih uporabljajo različne instrukcije različne načine naslavljanaja.

Združljivost. Dopuša se možnost uporabe vseh načinov naslavljanja pri kakršnemkoli tipu operatorja ali podatka. Npr., programski jeziki upoštevajo tip kot posebnost podatka, medtem ko večina strojev upošteva tip kot posebnost operatorjev.

Eden od vseh. Izvajanje je možno le na en sam način, ki se lahko izbere med vsemi možnimi načini. Npr. ali samo .EQ. in .LT., ali vseh 6 relacijskih operatorjev. Najbolj pogosto so trije ali štirje.

Podpora gradnikom, in ne integraciji. Problem ni v generirenju koda, ampak v kodni optimizaciji. Jezikovni stavki FOR in CASE so neuporabni.

Naslavljanje. Naslavljanje ne sme biti omejeno le na naslavljanje preprostih polj in zapisov. Večina strojev ne zagotavlja ustrezne podpore zapletenemu naslavljanju.

Podpora okolju. Podpora izvajalskemu okolju, ki še zdaleka ni tako dobra, kot je podpora aritmetiki in logiki, se mora povečati. Npr.,

Preveden program	C prevajalnik na VAX 11/780					C prevajalnik na RISC				
				VAX/RISC					VAX/RISC	
ime	obseg (vrstic)	na VAX (sek)	na RISC 8MHz	na RISC 12MHz	8	12	na VAX (sek)	na RISC 8MHz	na RISC 12MHz	8	12
ld.c	1587	27.9	21.0	13.9	1.3	2.0	35.2	22.4	14.8	1.6	2.4
sort.c	875	17.4	13.2	8.7	1.3	2.0	20.0	13.2	8.7	1.5	2.3
puzzle.c	118	5.2	3.6	2.4	1.4	2.2	7.3	4.8	3.2	1.5	2.3
skupaj	2578	20.5	37.8	25.0	1.3	2.0	62.5	40.4	26.7	1.5	2.3

Slika 3. Primerjalni test zmogljivosti C prevajalnikov na strojih VAX in RISC

skupinam skladov, dinamičnim povezavam, izjemnim situcijam ipd.

Odkloni. Odkloni od osnovnih principov morajo biti neodvisni od implementacije. Značilno je, da razvoj tehnologije vseskozi zaostaja za novimi arhitekturami.

Podpora materialni opremi v RISC-VLSI tehnologiji so HDL uporabniški programski paketi, ki so posebej namenjeni načrtvanju RISC-VLSI vezij. Poseben opisni jezik omogoča učinkovit opis tovrstnih procesorskih arhitektur. Pri MP2D pakiranju se uporabljajo ustrezna orodja za razmeščanje in povezovanje v postopku načrtovanja VLSI vezij.

Med najbolj pereče probleme načrtovanja RISC arhitektur je vprašanje kompleksnosti VLSI vezja, ki naj bi se razumno omejila. Z novo arhitekturo se poveča tako število kot kvaliteta vgrajenih novih virov, naraste pa tudi zanesljivost sistema. Vendar prinaša nova arhitektura tudi negativne vplive na gonilno sposobnost (moč) vrat, ki s kompleksnostjo vezja upada. Zakasnitve vsled dekodiranja in daljših prenosnih poti podatkov naraščajo, s tem pa se manjša hitrost CPU. Ti negativni učinki so še bolj izraziti pri CISC arhitekturi, kjer je kompleksnost VLSI vezja enega samega izredno sposobnega procesorja, nuja. S kompleksnostjo čipa se skokoma povečajo tudi stroški načrtovanja in testiranja procesorja. Zato je pričakovati rešitev le v multiprocesorskih sistemih z izkoriščanjem velike stopnje sočasnosti, medtem ko so posamezni procesorji v RISC-VLSI submikronski tehnologiji s sorazmeroma preprosto organizacijo zelo hitri in zato učinkoviti v izvajanju.

Iz gornjega je mogoče povzeti filozofijo RISC arhitekture:

1. Izbor tipa in količine virov mora maksimizirati pozitivne in minimizirati negativne učinke.
2. Učinkovitost stroja mora biti ovrednotena s časom izvajanja HLL programov in ne s številom strojnih instrukcij, ki pripadajo enemu samemu HLL stavku.
3. Pomembno je, da se kar najbolj poenostavi kodni optimizator, ne pa generator koda.

3. PREGLED RAZVOJNIH DOSEŽKOV

Razvoj RISC arhitektur je najpreje potekal v raziskovalnih laboratorijih firme IBM (računalnik z oznako 801), University of California at Barkelev (RISC I in RISC II) in Stanford University (MIPS). Glej sliki 1 in 2 !. Tri manjše firme so že realizirale produkte na osnovi Risc tehnologije, večina glavnih dobaviteljev računalnikov pa je pričela dobavljati tudi že prve poskusne serije komercialnih Risc računalnikov.

RISC I je bil implementiran v letu 1982, RISC II pa v letu 83. Oba imata 32-bitno besedo in 4 gigazložni virtualni adresni prostor. Razlikujeta se v tem, da ima prvi dvostopenjsko cevane instrukcije, 44k tranzistorjev in 500 nsek uro, medtem ko ima drugi trostopenjsko cevane instrukcije, 40k tranzistorjev in 330 nsek uro.

IBM je vsekakor pionir razvoja RISC računalnika. Razvil je miniračunalnik 801 z zmogljivostjo 2 MIPSa. Zasnovan je na emitersko sklopljeni logiki. Zasnova tega računalnika izhaja iz leta 1975, danes pa IBM že zakjučuje razvoj dveh vsebinsko povezanih različic komercialnih, na Risc tehnologiji zasnovanih računalnikov. Oba projekta sta poznana pod skupnim imenom ROMP. Eden od njiju je večuporabniški sistem za prevajalniško procesiranje, ki bo, sodeč po govoricah, v kratkem tudi komercialno dosegljiv.

Domnevajo, da bo zmogljivost tega računalnika med 2 in 10 MIPSov. Namenjen je tržišču superračunalnikov, ki so grajeni z 16 in 32 bitnimi računalniškimi čipi, podobnim procesorjema 68010 in 68020. Za novi računalnik je predviden operacijski sistem, ki je posebna verzija Unixa. Drugi projekt je na Riscu zasnovana inženirska delovna postaja, ki dela v območju nad 2 MIPSa.

Poleg omenjenih projektov obstajajo ob skupnem prizadevanju za čim tesnejše sodelovanje (npr. med IBM in Austin) še drugi projekti RISC računalnikov. Med njimi so različice računalnika 801, ki tečejo kot koprocesorji v sklopu nekaterih večjih IBM-ovih centralnih enot. TX je eden od rezultatov IBMovih naporov, da bi razvil napredno delovno postajo. Menda je že razvita RISC implementacija na nivoju vtične enote, namenjena računalnikom Serije 1 in PC. Ali bo rezultat teh projektov tudi komerecialno dosegljiv produkt, je nemogoče napovedati. IBM ima namreč navado, da razvija številne produkte, od katerih je marsikateri neuspešen. Zanesljivo je le to, da IBM ne bo hotel izgubljati prednosti, ki jo ima v novi tehnologiji, in bo še nadalje poskušal, da bo končno le uspel in čim več iztržil na račun že vloženih sredstev.

MIPS, Sun, Apollo, Silicon Graphics in Convergent so podjetja, ki planirajo prodajo kompletnih delovnih postaj. Njihovi RISC računalniški produkti na nivoju vtične enote (boarda) so predvsem zaradi optimiranih prevajalnikov kar desetkrat hitrejši od standardnih mikroprocesorjev.

Tudi DEC načrtuje rabo novih RISC konceptov. Danes že tečeta vsaj dva tovrstna projekta. Eden od njiju je znan pod imenom Nautilus. Računalniški produkt tega projekta v ECL tehnologiji bo imel zmogljivost nekaj 10 MIPSov in bo delno združljiv z VAXom. Cilj drugega pro-

jekta je inženirska delovna postaja, poimenovana Titan. Zmogljivost postaje bo 2 MIPSa in bo omogočala instalacijo nekatere VAXove aplikativne programske opreme, vendar z VAXom ne bo popolnoma združljiva.

Na konferenci pred dobrim letom dni je predsednik DECa Ken Olsen, ko je najavil VAX 8650 izjavil, da je bil DECov napor na razvoju RISC zasnovanih računalnikov doslej v večji meri neuspešen, hkrati pa dodal, da bodo kljub neuspehom tega razvoja še nadalje vlagali vse svoje napore.

Prav tako kot drugi se je tudi Hewlett-Packard oprijel nove tehnologije. Naznanil je, da bodo v bodoče vsi njegovi pomembnejši računalniški projekti zasnovani na RISC arhitekturi. Takšno usmeritev izvaja že pri nadaljni gradnji serije računalnikov HP 3000, ki bo združljiva z vso Hewlett-Packardovo obstoječo aplikativno programsko opremo. V okviru tega projekta, imenovan Spectrum, napovedujejo računalnik, ki bo zasnovan na več procesorjih s 64 bitnimi podatkovnimi besedami. Pričakujejo, da bo zmogljivost novega računalnika blizu 5 do 8 MIPSov, kar pomeni, da bo njegova zmogljivost napram računalniku današnje serije 3000 več kot podvojena.

Gigantski AT&T, ki dobavlja UNIX in C, napoveduje, da bo razvil na RISCu zasnovan "C stroj", ki bo optimiran za C jezik. Lastne in tuje razvojne grupe so temeljito ovrednotile rabo C programov, tako kot ld.c, sort.c in puzzel.c.

Tako je reševanje kompleksnih linearnih enačb v polni natančnosti na računalnikih RISC arhitekture pokazalo zelo dobre rezultate. Ena takšnih grup, kot je Argonne National Laboratory, je objavila rezultate testov, ki jih je opravila na 150-ih računalnikih z različnimi prevajalniki. Tabela na sliki 3 kaže primerjalni test zmogljivosti C prevajalnikov na strojih VAX in RISC.

Vse bolj je jasno, da RISC računalniki ne bodo le domena velikih firm, ki imajo vse vire za razvoj lastnih RISC računalnikov. Tudi manjši proizvajalci bodo svoje razvojne zmogljivosti uporabili za razvoj računalnikov - z vodili Multibus II in VME - na nivoju vtičnih kartic. MIPS Computer Systems Inc. od Mountain Wiev, Calif., je že razvil na nivoju vtične kartice in za VME vodilo RISC računalnik, ki je zasnovan na eni od prvih arhitektur, tj. na Stanfordovi MIPS arhitekturi. Zmogljivost tega računalnika sega do 8 MIPSov in je zgrajen okrog 32-bitnega RISC čipa, ki ima instrukcijske in podatkovne cache pomnilnike. Temu je dodan 64-bitni procesor za aritmetiko s pomično vejico procesne moči 10 MIPSov (kar pomeni miljon aritmetičnih operacij v pomični vejici na sekundo), ki je prav tako v RISC tehnologiji, ter odlično optimiran prevajalnik za UNIX V ali 4.2BSD. Vse to na eni sami vtični enoti.

Inmos, ena najpomembnejših britanskih firm računalniških komponent, je ponudila tržišču 16-bitne (T 212 in M 212) in 32-bitne (T 414 in kmalu tudi T 800) RISC procesorske enote, ime-

Operacija	Operandi	Komentar	Brezpog.indir.skok
Aritmetične in logične operacije			
Ada	R1,R2,Rd	Rd:=R2+R1	Celoštev. seštev.
And	R1,R2,Rd	Rd:=R2&R1	Logični "in"
Ic	R1,R2,Rd	Rd:=zlog R1 v Rd se zamenja z R2	Vstavitev zloga
Or	R1,R2,Rd	Rd:=R2VR1	Logični "ali"
RIc	R1,R2,R3,Rd	Rd:=R2VR3 rotiran preko R1 pozic.	Kombini.rotacija
RoI	R1,R2,Rd	Rd:=R2 rotiran preko R1 pozicij	Rotacija
SlI	R1,R2,Rd	Rd:=R2 pomak.levo preko R1 pozic.	Log.levi pomik
Sra	R1,R2,Rd	Rd:=R2 pomak.desno preko R1 poz.	Aritm.desni pomik
SrI	R1,R2,Rd	Rd:=R2 pomak.desno preko R1 poz.	Log.desni pomik
Sub	R1,R2,Rd	Rd:=R2-R1	Celoštev. odštev.
Subr	R1,R2,Rd	Rd:=R1-R2	Obrnjena Sub
Xc	R1,R2,Rd	Rd:=zlog R1 iz R2	Izločitev zloga
Xor	R1,R2,Rd	Rd:=R2(+)R1	Log.ekskluz. "ali"
Prenosne operacije			
Ld	A(R),Rd	Rd:=M(A+R)	Nalag.(baz.nasl.)
Ld	(R1+R2),Rd	Rd:=M(R1+R2)	Nalag.(baz.indeks)
Ld	(R1>>R2),Rd	Rd:=M(R1 pomaknjen preko R2)	Nalag.(baz.pomika)
Ld	A,Rd	Rd:=M(A)	Nalag.(nepos.nas.)
Ld	I,Rd	Rd:=I	Nalag.(takoj.nas.)
Mov	R,Rd	Rd:=R	Prenos (zloga/reg)
St	R1,A(R)	M(A+R):=R1	Shran.(baz.nasl.)
St	R1,(R2+R3)	M(R2+R3):=R1	Shran.(baz.indeks)
St	R1,(R2>>R3)	M(R2 pomaknjen preko R3):=R1	Shran.(baz.pomika)
St	R,A	M(A):=R	Shran.(nepos.nas.)
Nadzor prenosnih operacij			
Bra	Rd	PC:=Rd+PC	Brezpog.relat.skok
Bra	Co,R1,R2,Rd	PC:=Rd+PC če Co(R1,R2)	Pogojni skok
Jmp	Rd	PC:=Rd	Brezpog.skok nepo.
Jmp	A(R)	PC:=M(A+R)	Brezpog.skok baz.
Jump	A(R)	PC:=M(A+R)	Brezpog.skok posr.
Trap	Co,R1,R2	PC:=0 če (R1,R2)	"Trap" instrukcija
Druge operacije			
Save PC	A	MA(A):=PC(-3)	Reši večstop. PC po pasti/preki.
Set	Co,R,Rd	Rd:=-1 če Co(R,Rd) Rd:=0 če ni Co(R,Rd)	Pog.nastavitev

Slika 4. Strojni kod, ki je značilen za procesor z RISC arhitekturo

novane transputerje, ki so med seboj popolnoma združljive. Transputer je računalnik na čip, ki omogoča izvajanje več procesov hkrati in tudi sam skrbi za komunikacijo med njimi. Komunikacija poteka preko skupnega pomnilnika. Več transputerjev se lahko povezuje med seboj preko kanalov v večprocesorski sistem, ki omogoča konkurenčno izvajanje večih procesov. Takšne sisteme je moč še nadalje povezovati v še večje sisteme in tako graditi sisteme s poljubnim številom transputerjev. Transputerski sistemi niso več zasnovani na von Neumannovi arhitekturi, kakršno ima sam transputer. Zato, in zaradi sposobnega transputerja lahko dosegajo ali celo presegajo zmogljivost današnjih superračunalnikov. Zaključeni transputerski sistemi se povezujejo med seboj in s standardno mikroprocesorsko periferijo preko posebnih vmesnikov t.imenovanimi "link adaptorji" IMS C001 in IMS C002, ki skrbijo za medsebojno sinhronizacijo večih sistemov oziroma sistema z njegovo periferijo.

32-bitni transputer T 414 je splošno namenski in zmore 10 MIPSov pri 20 MHz. Prav tako je splošnonamenski njegov predhodnik T 212, medtem ko je M 212 namenski transputer za kontrolo inteligentnega diskovnega sistema. T 414, ki je predstavnik te družine, je izdelan v 1.5-mikronski CMOS tehnologiji s preko 150k transistorjev v 84-pinskem čipu. Procesor ima 32-bitne notranje in zunanje izhode za naslove in podatke, ki so multipleksirani in dosegajo hitrost prenosa 20 megazlogov. 4-gigazložni linearni naslovni prostor, PROM in 2k SRAM pomnilnika ter 4 medtransputerske komunikacijske kanale. V pogledu nabora ukazov transputer odstopa od običajnega nabora ukazov RISC arhitekture, predvsem po številu vseh ukazov in prisotnosti ukazov za množenje in deljenje.

Stavki	Pogostost uporabe	Stroine instrukcije	Pomnil. reference
Call/return	12	33	45
Loops	3	32	26
Assign	38	13	15
If	43	21	13
With	--	--	--
Case	1	1	1
GoTo	3	0	0

Slika 5. Pogostost pomnil.referenc v programih, ki so napisani v jeziku C ali Pascal

Stroj / procesor	čas izvajanja (%)	štev.izv. instruk. (%)	pomnil.dosto do podatkov (%)
VAX-11	26	5	19
PDP-11	22	19	15
68000	19	9	12
RISC 1	2	6	0.2

Slika 6. Primerjava procesorja RISC1 z nekaterimi drugimi procesorji

Proizvajalec transputerjev je poskrbel tudi za učinkovito in lahko programiranje v jeziku OCCAM, ki ga je Inmos posebej razvil za transputer. Prevedli so ga že tudi za druga okolja, npr. VAX in IBM PC.

NCR (Dayton, OH) je prva družba, ki je ponudila tržišču na RISCu osnovano vtično enoto za vodilo Multibus 1. Vtična enota NCR/32-796A je 32-bitni sistem, ki ima dva vmesnika za vhodne in izhodne podatke 24-bitnega Multibus vodila. Vtična enota je instalirana v razvojnem sistemu, ki je zasnovan na stroju NCR 3200, in dela ali kot koprocesor ali kot emulator zelo intenzivnega procesorja. Instrukcijski cikel je 150 nsek, ki izvaja prevajanje s hitrostjo 6.5 MIPSa. Nadaljni NECovi razvojni plani pa predvidevajo vtično enoto za 32-bitno vodilo.

Dandanes obstajajo trije komercialno dosegljivi računalniški sistemi, ki bazirajo na RISC arhitekturah. Ti sistemi so produkti firm Ridge Computers Inc. of Santa Clara, Calif., Pyramid Computer Inc. of Mountain View, Calif. in Celerity Computing Inc. of San Diego, Calif. Vse tri firme dobavljajo računalnike, ki so v razredu zmogljivosti najmanj 2 MIPSa. Ridge 32 ima poleg reducirane množice instrukcij tudi več prekrivnih registrov. Serijo Ridge 32 sestavljajo 32C, ki je večuporabniški sistem, in 32S, ki je majhen enouporabniški sistem, posebej načrtan za OEM uporabnike. Obe enoti imata 32-bitne podatkovne in adresne poti, 125 nsek cikel, cevanje instrukcij v štirih stopnjah, virtualni pomnilnik, procesiranje s plavajočo vejico in grafiko z bitno preslikavo visoke resolucije. Produkt je različica Berkeleyevega RISCa, posebej načrtovana za potrebe tržišča, le so: podpora materialne opreme hitremu kontekstnemu preklapljanju, ki dovoljuje rabo številnih vhodno/izhodnih naprav, več uporabnikov in procesov. Poleg tega uporablja Ridge 32 namesto enega običajnega VLSI čipa standardni TTL. Operacijski sistem je ROS, izpeljan iz UNIX V in 4.2BCD. Ta omogoča dodatne zmogljivosti, ki jih večina implementacij UNIXa nima, le so: pagemiranje virtualnega pomnilnika, visoko zmogljiv datotečni sistem in hiter medprocesni komunikacijski sistem, ki komunicira s sporočili.

Posebna odlika sistema Ridge 32 je učinkovita podpora grafiki. 19" barvni zaslon ima osemravninsko resolucijo 1024x768 pixelov, kontrolna vtična enota pa 128k lastnega pomnilnika za grafiko. Ta dopušča osveževanje 60 zaslonskih slik na sekundo. Podobno kot pomnilnik je paginiran tudi vhod in izhod iz diska. ROS ima večokenski, zaslonsko orientiran urejevalnik za učinkovito strežbo grafiki.

Računalniški sistem Pyramid 90x je 32-bitni računalnik z virtualnim pomnilnikom do 80 megazlogov. CPU s 125 nsek ciklom je nameščen na treh vtičnih enotah skupaj s procesorjem 68000 za sistemsko podporo in diagnostiko. Tako kot transputer tudi procesor Pyramid odstopa od klasičnih RISC kriterijev predvsem v dvocikličnih instrukcijah in v rabi bolj tradicionalne arhitekture za vhodno-izhodne operacije in specifičnega procesiranja. Zelo bogat je na registrih. 528 registrov je razvrščenih v 64 registrskih skupin. Te skupine registrov, ki imajo RISCu podobne prekrivne registre, podpirajo do 128 uporabnikov. Oba računalnika, Ridge in Pyramid sta zgrajena v MSI in VLSI logiki, medtem ko je Celerityjev računalnik C1200 zgrajen okrog NCR-ovega 32-bitnega procesorskega čipa, ki ima do 16 kilozložni registrski sklad.

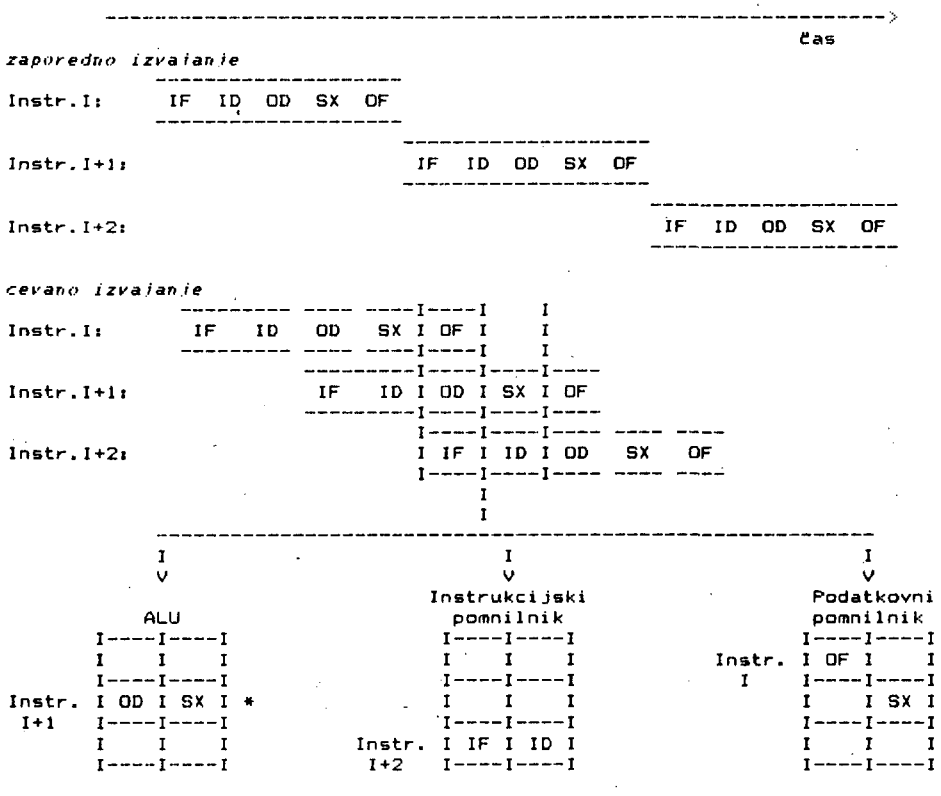
Pyramid 90x podpirajo HLL za CAD/CAE, podatkovne baze in expertni sistemi UNIX, operacijski sistem s časovnim dodeljevanjem, prevajalniki za C Pascal in F77. Dva do petkrat je hitrejši od VAX-11/780 pri mnogo nižji prodajni ceni.

4. ODLIKE RISC ARHITEKTURE

Risc računalniki so primerni za uvajanje številnih tehnik, ki so bile razvite že v preteklih letih z namenom, da se poenstavi in pospeši izvajanje programov, predvsem takšnih, ki so zapisani v visokonivojskih jezikih, in izvajanje funkcij, ki se nanašajo na operacijske sisteme. Johan Hennessey, ki je vodil razvoj danes že zastarelega Stanfordovega MIPS RISC procesorja, in njegovi komercialni nasledniki, so uporabljali pri opisovanju RISC arhitektur vzdevek "prodorna" arhitektura.

Zakaj takšno navdušenje nad RISC arhitekturami? Odgovor je preprost. Te arhitekture težijo predvsem k preprostejšim notranjim strukturam z namenom, da se poveča hitrost procesiranja in premosti silikonska bariera, ki zavira hitrejši razvoj čipa in večjo gostoto elektronskega vezja na mikroprocesorski čip. V drugem razdelku so bile že podane bistvene karakteristike RISC arhitekture, učinkovitost le-te pa se nazorno pokaže šele z zmogljivostjo že obstoječih sistemov z RISC čipi. Z njimi sta dosežena najmanj 2 MIPSa, kar je skoraj dvakrat več, kot zmore DECov VAX s procesorskim čipom 78032. Zaradi optimiranega izvajanja programske opreme in direktnega izvrševanja kode, je jasno, da ima RISC prednost v mnogih procesnih aplikacijah. Kljub temu pa je, in bo tudi v bodoče, še veliko računalniških sistemov, ki kode ne izvajajo direktno, vključno z upravljanjem programske prekinitve, dodeljevanja virov ter vhodov oz. izhodov.

Raba modernih instrukcij, za katere so značilne tudi preproste instrukcije tipa load/store (in ne takšne, kot so sicer potrebne za prenašanje blokov in podobno) in za katere je pomembno, da se izvajajo v enem ciklu, narekuje uvajanje drugačnih procesorskih čipov. Načrtovanje leteh je močno poenostavljeno. Mikroprogramiranje ni več potrebno in bistveno se zmanjša tudi kontrolna logika. Vzrok preprostosti RISC zasnovanega procesorja je v bistveno manjšem kodu asemblerskega jezika. Takšen kod, ki je zelo preprost, je prikazan na sliki 4 za MIPSov čip.



*) pomeni, da je ALU rezervirana za rabo OD in SX instrukcije I+1

DODELJEVANJE VIROV MED PREVAJANJEM PRI CEVANJU INSTRUKCIJ RAČUNALNIKA MIPS

Stanje	*Mnemonik*	*Opravila*
Dostava instrukcije	IF	*Dostava iz PC in PC inkrementira*
Dekodiranje instr.	ID	*Dekodiranje instrukcije*
Dekod. operanda	OD	*Izračun ef.adr.in nasl.pomn., če "load/store"* *Izračun nove vsebine PC, če razvejitev,* *sicer se uporabi ALU za reg-reg operacije*
Shran./izvajanje operanda	SX	*Vpis operanda, če "store"* *Raba ALU za komparacijo, če "compare-and-branch",* *sicer se uporabi ALU za reg-reg operacije*
Dostava operanda	OF	*Odčitanje opreanda, če "load"*

GLAVNE STOPNJE IN FUNKCIJE CEVANJA

Slika 7. Večina instrukcij se lahko izvaja konkurenčno s cevanjem

V nasprotju z zgoraj opisanimi značilnostmi pa dandanes večina procesorjev uporablja mikroprograme, ki omogočajo izvajanje vsake strojne instrukcije. Pri načrtovanju VLSI vezij postane zelo zahtevna ne le dekodirna logika številnih in zapletenih instrukcij, temveč tudi implementacija mikroprogramiranja. Nasploh je opazna tendenca, da postaja kontrolna logika neregularna in kompleksna ter zato zaseda že velik del silicijeve površine.

Arhitekture s številnimi registri so tudi manj zahtevne v pogledu materialne opreme, ki upravlja klice procedur in ukaze za vrnitev iz procedure v glavni program. Tovrstni upravljalni postopek potrebuje pri večini programov, ki so napisani v C jeziku ali v Pascalu, več pomnilniških referenc, kot kaka druga aktivnost. Iz tabele na sliki 5 je razvidna v procentih izražena pogostost uporabe posameznih stavkov, ter strojnih instrukcij in pomnilniških referenc v stavkih samih. RISC navadno uporablja (pushdown) sklad skupin registrov. Vsaki proceduri je dodeljena skupina registrov. Skupine se med seboj prekrivajo in zato so prekoračitve pri prenašanju parametrov manjše.

S kontekstnim preklapljanjem iz ene procedure v drugo (rešitev enega okolja, postavitev naslednjega) se nalaga oziroma prazni skupina registrov v skladu. Omenjena manjša zahtevnost registrsko usmerjenih arhitektur - v pogledu sicer že preveč obsežne materialne opreme - in učinkovitost takšnih arhitektur je nazorno prikazana s primerjavo procesorja RISC I z nekaterimi drugimi procesorji (Glej sliko 6!).

Registrsko usmerjene arhitekture zahtevajo bistveno manjšo pasovno širino pomnilnika, kot se zahteva za podatek v registrskih bankah na čipu samem. Nadalje se je izkazalo, da se z večanjem registrskih skupin ne veča tudi učinkovitost procesiranja. Pri mnogih aplikacijah je osem registrov na registrsko skupino dovolj za učinkovito procesiranje pri občutno manjši rabi pomnilnika.

6. VLOGA PROGRAMSKE OPREME RISC

Pri načrtovanju RISC arhitekture je ključnega pomena sistemska programska oprema, ki v RISC arhitekturi nadomešča sicer kompleksno materialno opremo CISC arhitekture. Med načrtovanjem materialne opreme, bodisi na nivoju kontrolne logike, bodisi na nivoju mikrokode, se načrtovalec soočamo s številnimi problemi izvajanja instrukcij. Pri RISC arhitekturi so ti problemi preneseni na programsko opremo in jih v večji meri rešujejo prevajalniki. Ne rešujejo se več med izvajanjem samim. Prednost tega je, da za specifične primere ni potrebno več vnaprej načrtovati izvajanje vsake instrukcije.

Omenjena problematika se med drugim pojavlja pri koordinaciji procesov, ki je izvedena z materialno opremo procesorja. Večina cevanih instrukcij procesorja se lahko izvaja konkurenčno, vsaka od njih trenutno na drugi stopnji izvajanja. Primer ponazarja slika 7. Koordinacija procesa preprečuje izvajanje tistega koraka izvajanja instrukcije, ki je odvisen od trenutno še nedostopnega rezultata izvajanja prejšne instrukcije.

Koordinacija procesov cevanja pri MIPS procesorju ni rešena z materialno opremo, kot je sicer običajno. Namesto nje skrbi za detekcijo koordinacije procesov prevajalnik in njegov poprocesor, imenovan reorganizator. Ta s primernim razvrščanjem instrukcij preprečuje vsakršne konfliktne situacije. Način koordinacije s prevajalnikom in poprocesorjem dokaj učinkovito zmanjšuje kompleksnost in obsežnost koordinacijske kontrolne logike.

Med največje rezultate naporov za povečanje učinkovitosti instrukcij štejemo uporabo razvejitev v obstoječih arhitekturah, ki izvajajo instrukcije s cevanjem. Z omenjenimi razvejitvami dodatne spremne instrukcije pri cevanju niso več potrebne, prisotne pa so manjše časovne izgube zaradi primerov ponovnega polnjenja cevi s tokom nove instrukcije. Izgube so lahko znatne le v takšnih aplikacijah, pri katerih dosega (kot na primer pri VAXu) čas izvajanja razvejitvenih instrukcij do 25% časa izvajanja vseh instrukcij programa.

MIPS je načrtan tako, da prevajalnik najpreje pregleduje instrukcije glede na razvejitvene izgube, nato pa jih ponovno razvrša s pomočjo t.im. tehnike zakasnenega razvejevanja (Glej sliko 8!). S to tehniko se lahko izkoristi do 90% najmanjšega možnega časa izvajanja.

MIPSova programska oprema, ki se nanaša na materialno opremo, pa se mora spopri jemati še z drugimi problemi. Eden teh je na primer tesno povezana prekoračitev majhnih rutin pri vstavljanju koda namesto klicev.

RISC, tako zaradi svoje navezanosti na programsko opremo, kot zaradi odtujitve mikroprogramskemu nivoju, dobro podpira viskonivojske jezike (HLL). Praviloma nima treh jezikovnih nivojev: viskonivojskega, zbirnega in mikrokodnega, ampak direktno preslika dan (običajno HLL) jezik neposredno v strojni kod, ki je z materialno opremo že tudi izvedljiv. In ne samo to, zaradi preproste arhitekture je število procesorskih instrukcij, ki so potrebne za izvajanje stavkov visokega programskega jezika, majhno.

6. RISC IN VLSI

Omejene razvojne možnosti, ki so pogojene s silicijem, kaže med drugim tudi zgodovina razvoja 32 bitnih procesorjev, kot so Nationalov 32032, Motorolin MC68020 in Intelov 80386. Pri vsakem od njih so bili številni spodrsljaji in težave, tako pri načrtovanju, kot tudi pri proizvodnji. Slika 9 kaže primerjavo značilnih podatkov o razvoju mikroprocesorjev RISC I in dveh novejših mikroprocesorjev.

Iz slike 9 so razvidne razlike pri načrtovanju logičnih planov (logic layouts). Za RISC je značilen nizek procent iregularne kontrolne

Naslov	Normalna razvejitev	Zakasnjena razvejitev	Optimiranje pri zakasnjeni razvejitvi
100	LOAD X,A	LOAD X,A	LOAD X,A
101	ADD 1,A	ADD 1,A	JUMP 105
102	JUMP 105	JUMP 106	ADD 1,A
103	ADD A,B	NO-OP	ADD A,B
104	SUB C,B	ADD A,B	SUB C,B
105	STORE A,Z	SUB C,B	STORE A,Z
106		STORE A,Z	

Slika 8. Razvrščanje instrukcij s tehniko zakasnjenega razvejevanja

logike in velik procent regularne logike, kot so registri. Regularna logika omogoča lažje načrtovanje, ki je zato tudi hitrejše. S tem se bistveno skrajša čas, ki je potreben, da steče redna proizvodnja produkta, računajoč ta čas od začetka njegovega snovnja.

RISC je zaradi svoje relativne preprostosti primeren tudi za namenske VLSI procesorje. Projektanti produktov, ki so namenjeni procesiranju visokonivojskih jezikov, so se hitro oprijeli RISC arhitektur. Projektov te vrste je več. Pri Berkeleyu sta zasnovana dva projekta. Prvi projekt predstavlja implementacijo Smalltalka na RISCovem čipu v enoti sistema na nivoju vtične kartice delovne postaje firme Sun Microsystems. V drugem projektu pa je RISCov čip namenjen izvajanju jezika Lisp in je vgrajen v večprocesorski konfiguraciji. Sistem omogoča gradnjo konfiguracije z največ osmimi kartičnimi enotami, osnovanimi na RISCu. Vsaka teh enot ima velik cache pomnilnik. Povezane so na vodilu skupaj s skupnim pomnilnikom, preko katerega tudi komunicirajo.

Razvojne karakteris. mikroprocesorjev	Zilog Z8000	Motorola M68000	RISC1
Vse pomnil. naprave	17.5k	68k	44k
Vse pomn.naprave brez ROM pomn.	17.5k	37k	44k
Pogonske naprave	3.5k	3.0k	1.8k
Regularizacijski faktor	5.0	12.1	25
Velikost čipa (mils)	238x251	246x281	406x305
Površina (kv.mil)	60k	69k	124k
Velikost kontrolne pov.(kv.mil)	37k	42k	7k
Kontrola v % od celot.površ.	53%	62%	6%
Trajanje razvoja do 1. silic.produk.(meseci)	30	30	19
Obseg razvojnega dela (človek mesecev)	60	100	15
Uvajanje v proizvodnjo (človek mesecev)	70	70	12

Slika 9. Primerjava mikroprocesorskih razvojnih karakteris. med RISC1 in njemu primerljivima procesorjema Z8000 in M68000

7. ZAKLJUČEK

RISC arhitekture se vse bolj uveljavljajo na določenih področjih, kot so npr. delovne postaje in namenski procesorji. Vendar pričakujejo, da bodo obstoječe arhitekture še precej časa nepogrešljive, saj bodo morale dalje podpirati obsežno programsko opremo in aplikacije, ki so bile razvite za številne mini in "maxi" računalnike. Tako ni pričakovati, da bi VAX ali PDP-11 kaj kmalu izginila iz prizorišča računalniške opreme.

Lahko pričakujemo, da bo uporaba RISC arhitekture vse pogostejša na novejših prodornih področjih aplikacij, medtem ko se bodo starejše arhitekture, kot je bilo že rečeno, še vedno uporabljale. Veliko tehnik, ki je bilo posebej razvitih za RISC arhitekture, se bo prilagodilo in vpeljalo tudi pri starejših arhitekturah. Mednje sodi predvsem programska optimizacija materialne opreme in raba arhitektur, ki so bogate na registrih. Kazalo je že, da RISC arhitektura procesorja ne bo sprejemljiva tam, kjer je pomembna izjemna hitrost, kot npr. v primeru operacij aritmetike s plavajočo vejico, hitrega vhodno/izhodnega procesiranja in v primeru podpore zahtevnemu večuporabniškemu procesiranju, ki je bogato na vhodno/izhodnem procesiranju. Razvojni dosežki v zadnjem času pa kažejo, da bo RISC prisoten tudi v takšnih primerih. To že danes zagotavlja najnovejši Inmosov procesor T 800, ki je najhitrejši 32-bitni mikroprocesor na enem čipu (1.5 MFLOPS pri 20 Mhz). Njegov komunikacijski protokol za serijske kanale omogoča 2.4 Mb hitrosti prenosa v obe smeri.

Verjetno bodo RISC arhitekture imele tudi odločilen vpliv na nove generacije računalnikov zaradi dejstva, ki je vse bolj deležno ostre kritike na račun starejših arhitektur; kritike na račun časa načrtovanja in proizvodnje tako zahtevnih produktov, kot sta na primer čipa Motorola MC68020 in Intel 80386. Nova arhitektura bo omenjeni čas občutno zmanjšala do takšne mere, da bo zopet sprejemljiv.

RISC arhitekture so mejnik na področju načrtovanja računalnika. Lahko bi rekli, da predstavljajo pravo eksplozijo na tem področju. Dandanašnji procesorji so namreč tako zapleteni, da je njihovo načrtovanje postalo vse bolj domena velikih dobaviteljev polvodniških in računalniških produktov. Z omenjenim mejnikom pa bo mnogim inženirskim organizacijam omogočen razvoj zelo zmogljivih procesorskih sistemov za namenske aplikacije. Novi produkti te vrste bodo tudi koristen medij za emulacijo obstoječih ali starejših procesorjev, ker je RISC instrukcije mogoče preslikati v programsko reprezentacijo drugih sistemov.

Z nastopom RISC arhitektur je nastopil trenutek, ko sta si programska in materialna oprema postala enakovredna v procesu načrtovanja računalnika. Primerov, da bo najpreje razvit procesor in šele kasneje tudi programska oprema zanj, ne bo več. Programska oprema bo postala integralni del razvoja materialne opreme.

Priložena obsežnejša literatura je namenjena bralcu, ki si želi pridobiti globlje znanje o obravnavanem, vsekakor perspektivnem razredu računalniških arhitektur.

7. LITERATURA

/ 1/ W.A.Wulf, "Compilers and Computer Architecture". IEEE Computer, July 1981.

/ 2/ D.A.Patterson, R.S.Piepho, "Assesing RISCs in HLL Support", IEEE Micro, Nov.1982

/ 3/ D.A.Patterson, C.H.Sequin, "A VLSI RISC", IEEE Computer, September 1982

/ 4/ M.G.H.Katevenis, "Reduced Instruction Set Computer Architectures for VLSI",University of California Technical Report,UCB/CSD/83/141, October 1983

/ 5/ Y.Tamir, C.H.Sequin, "Strategies for Managing the Register File in RISC,"IEEE Transactions on Computer, November 1983

/ 6/ R.W.Sherburn, "Processor Design Tradeoffs in VLSI". University of California Technical Report, UCB/CSD/84/173,April 1984

/ 7/ J.Hennesy, N.Jouppi, F.Baskett, J.Gill, "A VLSI Processor Architecture". In Proc. CMU Conference on VLSI Systems and Computations, pp.337-346, Computer Science Press, October 1981

/ 8/ J.Hennessy, N.Jouppi,J.Gill, F.Baskett, A. Strong, T.R.Gross, C.Rowen, J.Leonard,"The MIPS Machine ",In Proc.Compcon,pp.2-7,IEEE San Francisco, February 1982

/ 9/ J.Hennessy,N.Jouppi,S.Przybylski, C.Rowen, T.Gross, F.Baskett and J.Gill,"MIPS:A Microprocessor Architecture.In Proceedings of Micro-15, pp.17-22, IEEE, October 1982

/10/ J.Hennessy,N.Jouppi,S.Przybylski, C.Rowen, T.Gross, "Performance Issues in VLSI Processor Design, In Proc.Int.Conf.on Computer Design, IEEE, Rye, N.Y.,October 1983

/11/ J.L.Hennessy,N.Jouppy,F.Baskett,T.R.Gross, J.Gill,S.Przybylski,"Hardware/Software Tradeoffs for Increased Performance",In Proc SIGARCH/SIGPLAN Symposium on Arhitectural Support for Programming Languages and Operating Systems,pp.2-11,ACM,Palo Alto,March 1982, revised as Technical Report 82-228

/12/ J.Hennessy, N.Jouppi, S.Przybylski,C.Rowen T.Gross,"Design of a High Performance VLSI Processor". In Proceedings Third Caltech VLSI Conference, pp.33-54, 1983, available as Technical Report 83-236.

/13/ S.Przybylski,"Design Verification and Testing of MIPS, In Proceedings,Conference on Advanced Research in VLSI,p.100-109,Artech House, January 1984

/14/ S.Przybylski, T.Gross,J.Hannessy,N.Jouppi, C.Rowen,"Organization and VLSI Implementation of MIPS",Journal of VLSI and Computer Systems 1(3),Spring,1984, available as Technical Report 83-259

/15/ C.Rowen, S.Przybylski,N.Jouppi, T.R.Gross, J.Shott,J.Hannessy,"MIPS:A High Performance 32Bit NMOS Microprocessor",In Digest of International Solid-State Circuits Conf., IEEE San Francisco,Ca.,February 1984

/16/ T.R.Gross, "Code Optimization of Pipeline Constraints",PhD thesis, Stanf.University, August 1983, available as Technical Report 83-255.

/17/ J.Hannessy, "DARPA Research Review", Stanford University, December 26, 1984.

/18/ C.Barney,"RISC Technology Moves off Campus into Commercial Machines", Electronics Week, April 29, 1985

/19/ E.Basard, D.Folger, "Ridge-32 Archtecture-A RISC Variation", Proceedings of the IEEE ICCD '83, Port Chester, NY,October 31-Nov. 3, 1983

/20/ R.Ragan-Keller,R.Clark,"Applying RISC Theory to a Large Computer",Pyramid Technology Corporation Spec.Report on Minicomputer Systems,1295 Clarleston Rd, Mauntain View, CA 94043, January 10, 1985

/21/ T.Naegele, "Harris Goes After DEC with a RISC Architecture", Electronics Week, June 32, 1985.

/22/ B.C.Cole,"A Crowd of Hopefuls Warms up for 32-bit Microprocessor Race", Electronics Week, June 3, 1985

/23/ V.Milutinovic', "A Vertical-Migration Microprocessor Architecture", Purdue University Technical Report, TR-EE 84-36, August 1984

/24/ V.M.Milutinovic',"Risc architecture"(Tutorial) EUROMICRO 86, 12th Symposium on Microprocessing and Microprogramming, Venice, September 15-18, 1986

/25/ C.Barney, "DARPA Eyes 100 - MIPS GaAs Chip for Star Wars",Electronic Week, May 20, 85

/26/ "32-bit microprocessors and support devices" (product focus), Electronic Engineering, pp.110-141, October 1986

/27/ D.A.Patterson,P.Garrison,M.Hill,D.Lioupis, C.Nyberg, T.Sippel, K.Van Dyke, "Architecture of a VLSI Instruction Cashe for a RISC", Proceedings of the 10th ACM Conference on Computer Architecture,Stockholem, Sweden, pp. 108-116, June 1983

/28/ D.Ungar,R.Blau,P.Foley,D.Samples,D.Patterson, "Architecture of a SOAR: Smalltalk on a RISC",Proceedings of the 11th ACM International Conference on Computer Architecture, Ann Arbor, MICH.,pp.188-197, June 84

/29/ L.Foti, Et.Al., "Reduced - Instruction Set Multi-Microcomputer System",Proceedings of the NCC, Las Vegas, NE, July 1984

/30/ The IMS T 424 User Manuel, 1984
The IMS T 424 Technical Notes #1,#5,#6,and #8, 1984

/31/ R.Taylor, "Signal Processings with OCCAM and the Transputer,IEE Proceedings Part F, Vol.131, No.6, October 1984

/32/ I.Barron,et.al.,"Transputer doer 5 or More MIPS even when not used in parallel",Electronics,November 17,1983

/33/ P.Wilson, "Transputer", IEEE MIDCON, Sept. 1985

/34/ INMOS Transputer: Architecture (Reference manual), T414 transputer (Product data), C001 link adaptor(Product data), C002 link adaptor (Product data), September 1985

/35/ IMS P600 OCCAM Programming System VAX/VMS, Product description
OCCAM Portakit, Product description,
OCCAM Programming System, Product Overview
OCCAM Language Overview
OCCAM Select Bibliography
September 1985

/36/ C.Barney,"Supermini Has Stellar Performance", Electronics, August 11,1983

/37/ "Pyramid's New Supermini Uses RISC Technology",Electronic Engineering Times, August 13, 1983

/38/ R.Ragan-Kelley, "High -Level Language With RISC Support Makes a Fast Supermini",Research & Development, September 1984

/39/ R.Ragan-Keller,R.Clark,"Applying RISC Theory to a Large Computer",Special Report on Minicomputer Systems, Pyramid Technology Corporation, 1984

/40/ J.E.Smith,A.R.Pleszkun,R.H.Katz,J.Goodman, "PIPE:A High Performance VLSI Architecture Proceedings of the IEEE International Work shop on Computer Systems Organization, New Orleans, LA, March 1983

/41/ J.R.Goodman,J.-T.Hsieh,K.Liou,A.R.Pleszkun P.B.Schechter, H.C.Young,"PIPE: A VLSI Decoupled Architecture", Proceedings of the IEEE/ACM 12th Annual International Symposium on Computer Architecture, Boston, MA, June 1985

/42/ "Unix on the Ridge 32" , Ridge Computers, March 1984

/43/ Ridge Procesor Reference Manuale", Ridge Computers, June 1984

/44/ "Apple Benchmarks of the VLSI Technology Design Tools",Ridge Computers,August 1984

/45/ Nicol Mac, "Ridge Implements RISC - Based Personal Workstation",Digital Design,Oct. 1984

/46/ E.Basart,"How RISC Architecture Makes Maintenance-Type CAE/CAD Power in the Office Environment a Reality", Computer Technology Review, 1985

/47/ E.Basart, "Reduced Instruction Set Computers: Architectural Simplicity for Higher Performance at a Lower Cost",Computer Design, 1985

/48/ G.MacNicol, "A Risky New Architecture For The Future?", Digital Design, March 1985

/49/ D.May, R.Taylor, "OCCAM-an overview", Microprocesors and Microsystems,8,2, pp.73-79,1984

NEKATERE IZKUŠNJE PRI UVAJANJU PROLOGA V POUK RAČUNALNIŠTVA NA SREDNJIH ŠOLAH

UDK 681.3.06:319.682 PROLOG

Milan Meža
Center srednjih šol, Titovo Velenje

Pri pouku računalništva v srednjih šolah se učenci seznanijo le z enim od postopkovnih programskih jezikov, ne spoznajo pa novejših jezikov, ki so primernejši pri snovanju sodobnejših informacijskih sistemov. Pri programiranju le-teh se veliko uporablja programski jezik prolog. V članku so opisane izkušnje in preprosti primeri programov pri pouku tega jezika na srednji šoli v Titovem Velenju.

1.Uvod

Po učnem načrtu je pri pouku računalništva v srednji šoli predpisan eden od postopkovnih programskih jezikov.V preteklosti je bil v uporabi fortran, sedaj pa je priporočen pascal, ker omogoča lepši stil programiranja.Zaradi velike popularnosti basica na mikroračunalnikih, so se nekatere srednje šole odločile za uporabo tega jezika pri pouku računalništva.
Za programiranje sodobnih informacijskih sistemov pa postopkovni programski jeziki niso primerni.Za programiranje takšnih sistemov se največ uporablja prolog. Prolog je programski jezik, ki se močno razlikuje od drugih konvencionalnih,postopkovnih programskih jezikov. To je jezik logičnega programiranja, zato je potrebno korenito spremeniti način razmišljanja o problemu ter stil programiranja. Za delo na mikroračunalnikih je bil razvit mikro-prolog,ki se le malo razlikuje od prologa. Mikro-prolog je nepostopkovni jezik, zato se lahko pri programiranju osredotočimo na problem in ne na algoritem,po katerem bi problem rešili.Algoritem je vgrajen že v jezik.Prolog je izjemno učinkovito orodje za simbolično, nenumerično procesiranje, manj pa je primeren za numerično računanje. Je glavni programski jezik za programiranje sistemov umetne inteligence.Veliko je pridobil na popularnosti, ko je bil izbran kot osnovni jezik pete generacije računalnikov.Kernel Language Zero (KLO) je strojni jezik prvega računalnika pete generacije.Ta jezik je razširjeni prolog.

2.Uvajanje učencev v programiranje v prologu

Na Centru srednjih šol v Titovem Velenju smo uvedli na naravoslovno matematični usmeritvi pri pouku računalništva poleg pascala tudi mikro-prolog.Učenci izvajajo vaje na mikroračunalnikih DIALOG z LPA mikro-prologom verzije 3.1 in urejevalnikom teksta Word-Star.Pri učenju jim je v pomoč učbenik za mikro-prolog,napisan za interno uporabo na našem šolskem centru.Učenci so način logičnega programiranja dobro sprejeli in kažejo zanj večje zanimanje kot za programiranje v pascalu. Ko učenci spoznajo gradnike in sintakso mikro-prologa,začnejo z gradnjo svojih informacijskih sistemov.Informacijske sisteme gradijo za različna področja kot so geografija,kemija,matematika itd. ter jih pri vsaki uri dograjujejo.Takšni sistemi so zgrajeni iz podatkovnih baz in pravil, ki omogočajo učinkovito iskanje najrazličnejših podatkov iz teh baz.Poglejmo primer informacijskega sistema iz področja geografije.
V podatkovni bazi imamo državo, njeno glavno mesto ter geografske koordinate glavnega mesta.Podatkovno baza lahko vpišemo direktno:

```
((lokacija London Anglija 51 0))
((lokacija Rim Italija 41 -12))
((lokacija Madrid Spanija 40 3))
((lokacija Moskva SZ 55 -37))
((lokacija Tokio Japonska 35 -139))
    .
    .
```

Enostavneje pa je,če nas pri vpisovanju vodi računalnik po naslednjem programu:

```
((vpis)
 (PP Program za vpisovanje )
 (P Vpisi gl_mesto:)(R X)
 (P Vpisi drzavo:)(R Y)
 (P Vpisi zemljepisno sirino:)(R Z)
 (P Vpisi zemljepisno dolzino:)(R Z1)
 (ADDCL ((lokacija X Y Z Z1)) ))
```

Črke x,y,z,X,Y,Z, ki so lahko tudi indeksirane, so v mikro-prologu spremenljivke.Imajo lokalni doseg-svojo vrednost ohranijo le znotraj enega pravila.Do rešitve zastavljenega vprašanja pridemo s postavitvijo iskalnega vzorca, v katerem na neznana mesta postavimo spremenljivke,ki se jim priredijo ustrezne vrednosti iz podatkovne baze.Če postavimo vzorec ((lokacija x y 41 -12)),se bo spremenljivki x priredila vrednost Rim, spremenljivki y pa Italija.Če želimo iz podatkovne baze geografske koordinate želenega glavnega mesta, napišemo naslednje pravilo:

```
((koordinate x y z)
 (lokacija x z1 y z))
```

Torej: Mesto x ima koordinati y in z, če ima to mesto lokacijo y in z.

Naslednje pravilo poišče iz podatkovne baze glavno mesto podane države. Ker so inteligentni informacijski sistemi sposobni reševati probleme v obe smeri (vhod-izhod ali izhod-vhod), nam isto pravilo najde tudi državo danega glavnega mesta.

```
((gl_mesto x y)
 (lokacija x y z z1))
```

x je glavno mesto države y, če ima mesto x lokacijo v državi y.
Naslednje pravilo nam iz podatkovne baze poišče tista mesta, ki so severneje od podanega mesta.

```
((severneje x y)
 (lokacija x z1 X Z)
 (lokacija y z2 Y y1)
 (LESS Y X))
```

Mesto x je severneje od mesta y, če ima x zemljepisno širino X, mesto y pa Y in je Y manjši od X.
Mehanizem, ki najde vsa mesta, ki so južneje od mesta y je še enostavnejši, ker imamo že definirano pravilo za severneje.

```
((juzneje x y)
 (severneje y x))
```

Mesto x je južneje od mesta y, če je mesto y severneje od mesta x.

Za komunikacijo s sistemom uporabljajo učenci poleg vgrajenega pravila za spraševanje ? še pravili KATERI in JE ,ki ju sami sestavijo.

```
KATERI(x(gl_mesto x Italija))
```

Odgovor je :
Rim

Ali:

```
JE((gl_mesto Rim Italija))
```

Odgovor je:
DA

```
KATERI(x(severneje Rim x))
```

Za vstavljeno podatkovno bazo je odgovor:
London
Moskva

Čim obsežnejšo bazo in pripadajoča pravila zgradijo,tem več vprašanj je sistem sposoben odgovoriti.
Podobno lahko učenci sestavijo podatkovno bazo iz področja kemije. V takšno bazo vnesejo podatke za elemente iz periodnega sistema ter formule za spojine. Ko bazi dodajo še pravila za iskanje,lahko iz sistema dobijo najrazličnejše podatke.Enostavnost tako oblikovane baze kaže naslednji primer:

```
((element zveplo 16 S 32))
((element aluminij 27 Al 13))
((element baker 29 Cu 64))
((element kisik 8 O 16))

((formula voda (H 2 O)))
((formula m_galica (Cu S O 4)))
((formula glinica (Al 2 O 3)))
```

Bazi dodamo še pravila za iskanje po bazi:

```
((simbol x y)
 (element y z x X))
```

To pravilo omogoča iskanje simbola za podani element oziroma imena elementa za podani simbol.

```
((vsebuje x y)
 (formula y z)
 (clan X z)
 (simbol X x))
```

Pravilo 'vsebuje' nam izpiše imena elementov x, ki jih vsebuje spojina y.

```
((masa x y)
(OR((element x z z1 y))((element z z1 x y))))
```

S tem pravilom dobimo maso elementa,če imamo znano ime ali simbol oziroma dobimo ime in simbol elementa,če imamo podano maso.

```
((st_elektronov x y)
(OR((element x y z z1))((element z y x z1))))
```

Atom elementa x ima število elektronov y, če ima element x vrstno število y ali pa tudi,če ima element s simbolom x vrstno število y.
Poglejmo si nekaj vprašanj,ki jih lahko postavimo sistemu:
Kakšno formulo ima voda?

```
KATERI(x(formula voda x))
(H 2 O)
```

Katera spojina ima forulo CuSO4?

```
KATERI(x(formula x (Cu S O 4)))
m_galica
```

Kakšen je simbol za kisik?

```
KATERI(x(simbol x kisik))
O
```

Koliko elektronov vsebuje atom kisika?

```
KATERI(x(st_elektronov kisik x))
16
```

Katere elemente vsebuje voda?

```
KATERI(x(vsebuje x voda))
vodik
kisik
```

Koliko atomov vodika je v molekuli vode?

```
KATERI(x(formula voda (z x y)))
2
```

Kaj je poleg bakra še v m_galici?

```
KATERI(x(formula m_galica (ydx)))
(S O 4)
```

itd.

Čeprav prolog ni namenjen za programiranje problemov z veliko aritmetičnih operacij, pa lahko učenci preprostejše primere s tega področja programirajo tudi v tem jeziku.Naslednji program je namenjen za računanje volumna teles.

```
((izbor)(P "ČA")
(PP ********* VOLUMEN **************)(PP)(PP)
(PP 1.........volumen valja)(PP)
(PP 2.........volumen kvadra)(PP)
(PP O.........konec)(PP)
(PP Pritisni stevilko !)(PP)
RCLEAR
(RDCH "KBD:" Y)
(IF (EQ Y "1")((valj))((nic)))
(IF (EQ Y "2")((kvader))((nic)))
(IF (EQ Y "O")((konec))((nic))))
```

```
((nic))
((konec)(PP)(PP K O N E C))
((cas)(PP)(PP Pritisni RETURN!)
 RCLEAR
 (RDCH "KBD:" Y))
((produkt (x y) z)(TIMES x y z))
((produkt (xdy) z)(produkt y X)(TIMES X x z))

((valj)(PP Izracun volumna valja)
 (PP Podaj polmer : )(R X)
 (PP Podaj visino : )(R Y)
 (produkt (3.14 X X Y) Z)(PP)(P Volumen = )
 (PP Z)(cas)(izbor))
((kvader)(PP Izracun volumna kvadra)
 (PP Podaj rob a : )(R X)
 (PP Podaj rob b : )(R Y)
 (PP Podaj visino v :)(R Z)
 (produkt  (X Y Z) x)(PP)(P Volumen = )(PP  x)
 (PP)(cas)(izbor))
```

Program deluje podobno kot programi,ki jih pišemo za reševanje takšnih nalog v pascalu.Podan je zato,da lahko primerjamo program v mikroprologu s paskalskim,ki je večini programerjev bolj domač.Ko izberemo telo in podamo zahtevane podatke, dobimo izračunani volumen.

3.Zaključek

V članku so opisane izkušnje pri uvajanju logičnega programiranja z mikro-prologom pri pouku računalništva v srednji šoli.Podani so preprosti primeri programov,ki jih učenci z velikim zanimanjem pri pouku sestavljajo. Ker je programiranje v logiki način programiranja v prihodnosti, je nujno, da se takšen način razmišljanja za reševanje problemov vnese kot stalna oblika izobraževanje računalništva v srednje šole,ki so naravoslovno usmerjene, saj bodo takrat, ko bodo te generacije na višku ustvarjalnosti, postopkovni programski jeziki le malo v uporabi.

4.Literatura

K.L.Clark F.G.Cabe J.R.Ennals	Mikro prolog primer Sinclair Rsearch Ltd 1983
Ennals R.	Begining micro prolog Imperial college of Science and Technology University of London 1984
M.Meža	Gradnja različnih baz podatkov in znanja ter razvoj pripadajočih mehanizmov sklepanja v rač.logičnem jeziku mikro prolog Raziskov.naloga ORS Velenje 1986
I.Bratko V.Rajkovič	Računalništvo učbenik 1984

COMPUTATION MAY HINGE ON BIOLOGICAL MATERIALS

UDK 681.3:57

Sonja Jeram
Iskra Delta, Ljubljana

Abstract. Researchers are searching for new materials to construct a better computer. They are giving attention to biological materials such as bacteriorhodopsin, cytochrom C, porphyrins and DNA. A model of a molecular computer has already been constructed. It is a computer based on enzymes and its program is represented in the structure of proteins. That is why the programming depends on evolution by variation and selection. It is just one of the many attempts to construct an intelligent machine.

0. Introduction

There are two fundamental approaches that are both divergent and complementary. The first, motivated by biology, seeks to duplicate the powerful information processing capabilities of biological systems, such as pattern and object recognition, self-organization and learning, and effective use of parallelism - precisely the abilities not being mastered by today's computers and robotics. The second approach, motivated primarily by existing computer circuitry, seeks to create a molecular electronics that would lead to a smaller, faster, less expensive digital computer with increased memory density.

1. Classical Versus Molecular Computing

The important feature of this new way of computing is that it is context-dependent - that is, inputs are processed as dynamic physical structures (molocular computing) rather than bit by bit (classical computing). Because of this context dependence, molecular computers are particularly well-suited for processing patterns of sensory inputs, such as light, temperature, and pressure.

Molecular computing is a technology that uses physical recognition for computation. And indeed, for now, a major drive is to develop a intelligent preprocessor, a sensor that sees patterns, feels surfaces, and senses chemical gradients, and that feeds the resulting information to conventional computers for bit by bit processing.

In the classical computing machine paradigm, the human programmer conceives an algorithm, or a method for solving a problem, and codes it as a string of symbols. The string, or program, may act on data from the environment - for example, patterns of electrons that code sensory information. The program and data control the behavior of the machine, usually by using a compiler to code the program into the physical state of the machine

Three facts about this well-known process stand out. First, the machine is structurally programmable, since the program it executes is represented in its physical structure. Second, the machine computes symbolically, since it performs purely formal operations on physically distinct patterns. Third, creation of the program depends on human intelligence. So far as is known, it is possible to simulate any process in nature with symbolic computation, but to simulate the intelligence of the programmer would strain even the most powerful digital machine.

These three facts are modified in biomolecular computing. The fundamental switches are protein enzymes, and the manner of processing patterns is physical and dynamic rather than symbolic and passive. The program is implicitly, rather than explicitly, represented in the structure of proteins and of the system into which they are integrated. Therefore, structural programmability does not carry over; rather, programming depends on evolution by variation and selection.

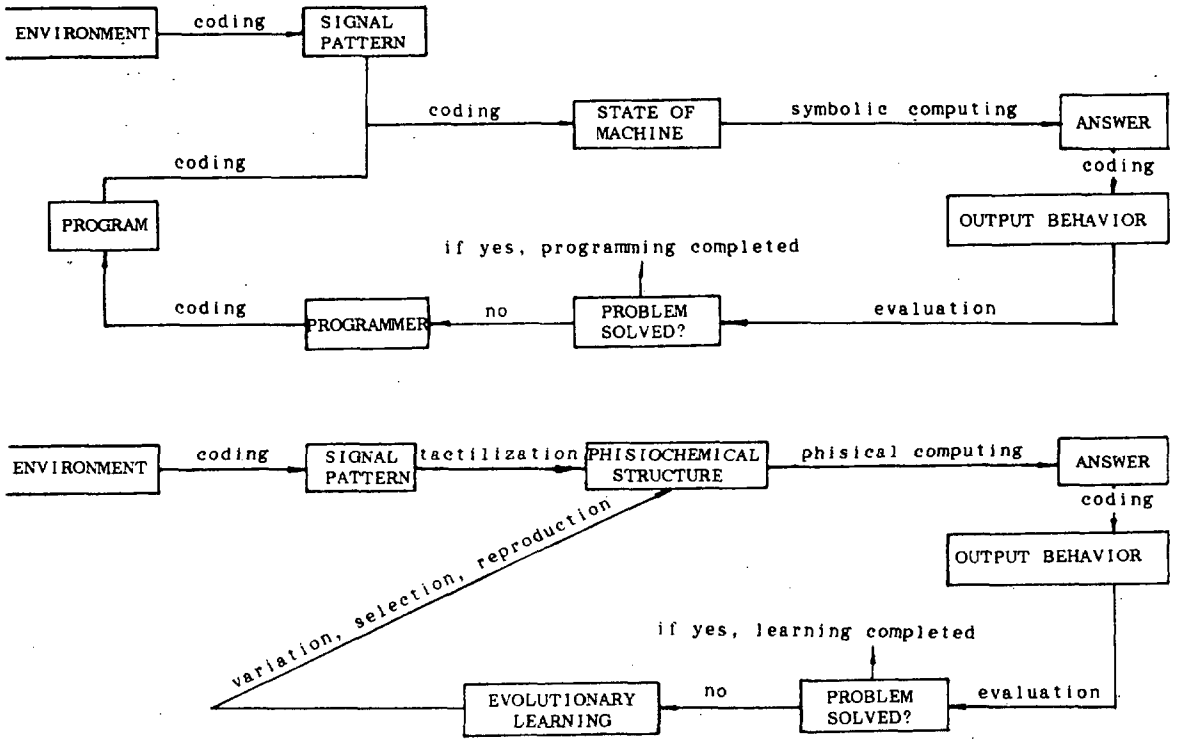


Fig. 1. Classical and biomolecular computing. In classical computing, the machine is programmable, it computes symbolically and the program depends on programmer. In the biomolecular computer, programmability is not possible, the processing is dynamic and programming depends on evolution by variation and selection.

2. Tactilizing Processors

A molecular computer design must solve the problem of exploiting geometry-recognitional capabilities of enzymes to process nontactile input signals, like photons or electric pulses. To do this, it will be necessary to embed the enzymes in a processor that "tactilizes" the input signal - that is, converts it into molecular and physiochemical forms that enzymes recognize.

Such a tactilizing processor integrates three layers of information processing. Receptors in the first layer transduce arriving signals into an "activity pattern" in the second layer. If the second layer consists of substrate molecules in a solution, the activity pattern is a reaction-diffusional pattern of these molecules. Enzymes in the third layer read out local concentration that results from the reaction, and uses the information to control the output signal emitted by the processor. The processor will have intrinsic generalization capabilities if the reaction-diffusional dynamics are not overly sensitive, since similar input patterns will then be treated like similar initial conditions.

The "program" of an individual processor is determined by the physics of the tactalization dynamics and of the readout enzymes. Unlike the minimal pattern processors in a classical *machine, these do not conform logically to any* abstract description. The programming of the molecular computer relies primarily on the evolutionary method of varying the readout enzymes or the tactilization dynamics, evaluating performance, selecting the best performing processors or replication, and repeating the cycle until the desired performance is obtained.

Tactilizing processors owe their efficiency to massive parallelism at the level of molecular activity, rather than to the speed of individual switching processes. Thus, the world of computing is divided into two radically different domains. The classical world achieves programmability at the expense of evolutionary adaptability and computational efficiency. The biomolecular world is not programmable, but compensates with evolutionary adaptability and increased efficiency.

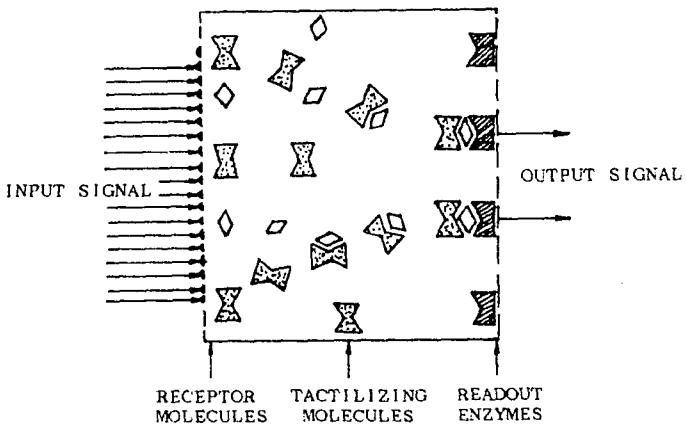


Fig. 2. Tactilizing processor. Three layers of biomolecules provide intrisic generalization capabilities. of the processor.

3. Molecular Possibilities of Implemetation

To implement this concept of a molecular computer it is necessary to search for suitable materials. The supporting technologies might be of a great importance.

Gene and protein engineering are central to tailoring proteins for the receptor and readout layers of a molecular computer, as well as proteins that use the self-assembly principle for self-fabrication and self-repair.

Immunological techniques, especially the modern technique of monoclonal antibody formation, make it possible to produce large quantities of identical immunoglobulins. This technique could provide a rich source of building blocks for molecular computers. Membrane engineering has the goal to attach active groups, like dye chromophores, to long chain hydrocarbons and to embed these in the membrane in a sufficiently ordered way to allow for a useful exchange of electrons, protons, photons, or excitons; a similar energy process enables plants to conduct photosynthesis. There are also other interesting technologies like polymer chemistry and dissipative structures.

4. A Model of Molecular Processor

One example of a molecular computing device comprises a bilayer membrane divided into compartments by a Teflon grid. Light signals falling on a pigment molecule would produce a potential change that would diminish with distance. Whether a readout compartment produced an output would thus depend on the level of excitation in neighboring compartments. This is analogous to the reaction-diffusional model of the neuron, except that it operates in microseconds rather than miliseconds. Some investigators have reported that bacteriorhodopsin can be stabilized in a membrane for up to a year.

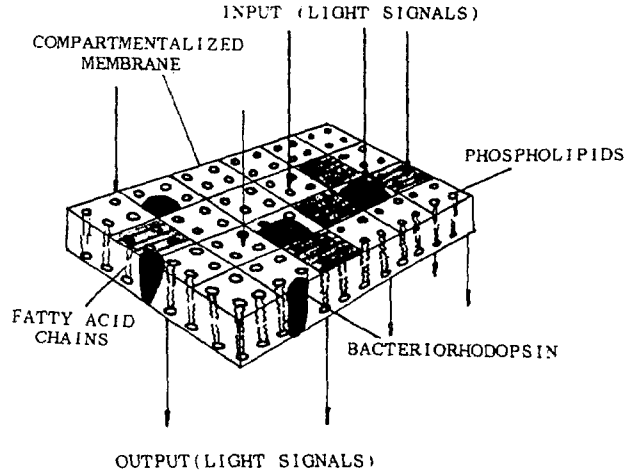


Fig. 3. Molecular computer device. Bacteriorhodopsin molecules in bilayer membrane receive the light signal and produce a potential change that is read by readout compartments which give the output signal.

5. A Survey of Research and Development of Appropriate Biomolecules

Many scientists are now searching for suitable materials and technologies that would provide molecular computing.

Mitsubishi Electric Corp.'s Central Laboratory has established techniques for growing protein film with molecules oriented for stable, unidirectional electron transmission. They used cytochrom C molecules to form the oriented molecular film using a laser beam technique.

Researchers at Carnegie-Mellon University are working on a high storage device and a fast NAND gate based on biological materials extracted from a bacteria. One such material is Bacteriorhodopsin. This molecule has an advantage in being highly stable and able to withstand high temperatures and optical radiation.
Bacteriorhodopsin is also investigated in Japan. It is known for its action as a proton pump under light irradiation. It has a cis-trans switching function and its practical application is anticipated for memory and switch devices.

Britain spends about US $ 7,9 million annually on research that falls into molecular electronics orbit. Researches are investigating the electrical properties of metallo-organic materials, such as porphyrins or phthalocyanines which change when they react with certain gases. The materials are used to build chemical sensors in which conventional chips sense the varying electrical output. Researchers in molecular electronics believe that these materials could be used to make organic equivalents to transistors.

6. Conclusion

Many biologists are also working on this problems. They try to find out if there is any possibility to apply, for example, recombinant DNA to microelectronics. But their main purpose is to construct a model of the molecular computer to know better how our own living system works.

In searching of new concepts, human is trying to build an intelligent molecular computer to further and advance the knowledge of its own intelligence.

References

Conrad M., The Lure of Molecular Computing, IEEE Spectrum, Oct. 1986, p. 55-60.

Mitsubishi one step closer to producing biodevice, Semiconductor International Magazine, May 1986.

Biological chips can store more, Electronics Weekly, 1 June 1986.

Pressure to spend more on biochip, New Scientist, Asia-Pacific Tech. Monitor, Nov.-Dec. 1985.

Bioelectronics in Japan, Science and Technology in Japan, Jan.-March 1986.

Mazur L., Electronics comes to life (biotechnology), Technology (GB), 29 Nov. 1982.

REPORT OF A JOURNEY

Parsys Expeditions to New Worlds (*)

Anton P. Zeleznikar
Iskra Delta, Ljubljana

0. The Parsys' Background

But he who is hated by the people
as a wolf is by the dogs:
he is the free spirit, the enemy of fetters,
the non-worshipper, the dweller in forests.

F. Nietzsche, TSZ, 126.

In ancient times, expeditions were sent to deserts and to far and unknown countries; they were instructed to discover, reveal, and conquer new land and people, new manpower, new goods, and also new cultures for the rulers and - after some time - also for the benefit of the rulers' populations. Nowadays, expeditions are going to universities, research institutes, and symposia to reveal, confirm, and obtain information concerning the future possibilities of development and survival for populations of hightec companies. In such expeditions, in addition to the intelligent actors - the expedition's main players - there have been reporters, silent observers, investigators, and recognizers of such marches, advances, and movings into new directions.

In the Parsys expedition, I was mostly a helpless, doubting actor; however, a sensitive observer, a systematic investigator, and last but not least, a strategic recognizer. Because, in accordance to my life experiences, my freelance and individualistic attitude, my scientific and philosophical orientation, my engineering work, my pedagogue (dogmatic and didactic) acting, my elapsing life stream, I have to simultaneously be inwardly on the sites of action (intelligent throwness) and observation (meditative thinking).

A long time ago, I decided to stay open, outside of any gentile formation, any particular ideology, any friendly brotherhood, professional or scientific fraternity. Not because of a rigid life principle (which would be an artificial construction), but to insure the possibility of acting spontaneously and because of my love for freedom. This orientation, qua my own experience, showed me simply how progress and chance can proceed in complex and unforeseeing undertakings - such as the Parsys project undoubtedly is. How to mobilize and how to bring into a freely directed movement certain innovative and individual, scientific, engineering, technological, and philosophical workers, in order to implement a large project? How to offer them a common objective and simultaneously preserve their individual motivations? How to organize and connect them to the development-essential information and into co-operation with advanced (technologically and philosophically avant-garde) research teams all over the world (U.S.A., Japan, Western Europe, etc.)? How to keep their mental and physical condition to sustain on the innovative way with their full and free interest?

Quite at the beginning, Iskra Delta had introduced and incorporated strategic thinking and acting in its managerial decision making. As a fast growing enterprise in the field of computer industry, it has been confronted not only with the very basic organizational and technological problems of modern computer industry of the developed hemisphere, but also with specific problems of a technologically and even civilizationally (socially, ideologically) underdeveloped environment. In these times, up to this day, the general director of Iskra Delta - Mr. Janez Skrubej - was not only the real strategist of the company, but also the optimistic fighter, organizer, believer of the progress, and the carrier of several hard, arduous, and exhausting business situations and developmental processes of the company. And all of this in irregularly and unforeseeably changing circumstances of the underdeveloped hemisphere. Thus, it is to say, that he was the main initiator and supporter of the innovative and independent Parsys project.

What I am trying to explain from my observational point of view is that the background of the Parsys undertaking can be understood only through the overseeing of the roots and specific conditions in which the project arose. Mr. Petar Brajak, whose experience was the most valuable contribution to the beginning of the project, was the first actor and conquerer in the generally unknown conceptual area of the project. Professor Saša Prešern was the first commander and organizer of the project's undertakings. My own role was mostly observant, propagating philosophical and futuristic brightness into the project's perspective. It can be said that starting of the Parsys project happened outside of any traditional scientific, developmental and organizational experience and - to all of this - in the era of advancing and all-embracing, general social and intellectual crisis.

1. Early Philosophy of the Parsys Project

Whence things have their origin,
there they must also pass away
according to necessity;
for they must pay penalty
and be judged for their injustice,
according to the ordinance of time.

The Anaximander Fragment (F. Nietzsche)
in Heidegger, EGT, 13.

Up to now, I did not explain the meaning of the identifier Parsys. It could be pretty well the name of a mysterious being from the ancient Greek mythology. However, Parsys stands for Parallel computer system. Its origin, as the Iskra Delta identifier and a possible future product, has to be sought in concepts coming from the Japanese Fifth Generation initiative and from some other innovative projects of parallel computer architectures over the globe.

The conception of the Parsys philosophy was never only architectural. The question was how to bring the problems of applicability, user-friendness, marketability into the

---

Figure 1. Mr. Takashi Kusanagi, the president of Ampere, during our second trip to Japan in June, 1986.

Figure 2. From right to left: Mr. Ryu Osaki, the director for international operations; Mr. Takashi Kusanagi, and the author at lunch in the Sunshine Hotel in Tokyo (June, 1986).

Figure 3. Professor Eiichi Goto from the Tokyo University. In June 1986 we met again after a 15 year period.

Figure 4. Professor Eiichi Goto had the most attractive introductory speech at the 13th Annual International Symposium on Computer Architecture, presenting his project of the ultra-fast supercomputer.

Figure 5. The time came when we could speak in peace about unusual technologi-cal, scientific, and our own life-concerning problems, how to visit the very few people who remain to be of our interest throughout the rest of our lives. This was the time at professor Goto's laboratory at Tokyo University.

architectural context and also how to draw the necessary motivation of integrative power (national and economic interest, manpower for research and development, international connectiveness, financial investment, etc.) into the philosophy of the project. In this respect, two former investigational trips to Japanese universities, institutes, companies, and symposium have not been incidental.

In November, 1985, during my first visit to Japan, I was mainly the observing listener in the laboratory of professor Toshio Sata at the Tokyo University. Professor Sata was the first one who revealed to me the strategic thinking of the Japanese academic and governmental intellectuals, to the consequences and actions which followed from such a bethinking. In this way, he was explaining the national strategy of ICOT (Institute for New Generation Computer Technology) in the most didactic way, stressing the importance of the Japanese intellectual capabilities and their concentration. With this concept, ICOT's activity looks like a wedge which progresses into unexplored technological and conceptual fields of future computer systems and which, through its organization, government's and enterprises' support takes care of the propagation of its activity and of the common investigational results in the broad industrial environment. This is the way ICOT is solving the problems of fast drawing the Japanese industry into new technological regions.

In Riken (the Institute for Physical an Chemical Research in Saitama) I have learned how a small group of highly skilled individuals can make a real contribution to the most pretentious scientific, technological, and industrial products. In the Eiichi Goto's Laboratory for Information Science, a series of original research was performed which resulted in the development of top-technological equipment (e. g. original Lisp machine, named FLATS; electron beam exposition system for down to 0.1 micron technology; etc.) A demonstration of this equipment convinced me that scientific and technological achievements depend mostly on a motivating orientation and on the highly intellectual individual capabilities of researchers. So, I could explain the lack of scientific and technological achievement in my domestic environment. At that time, I stated the question, how a group of 6 researchers in Saitama, in the last 10 years, has contributed much more to the world's technological progress than my previous, similarly structured institute in which 600 people worked (STL).

In Japan, we had several interesting demonstrations of our VME Intel processor board (of Iskra Delta's Trident System) at some Japanese companies (Bug, Sapporo and Ampere, Tokyo). The president of Ampere, Mr. Takashi Kusanagi, had also arranged my visit to ICOT; it was not possible to arrange this visit with the help of different professors from the Hokkaido University in Sapporo. At Hokkaido University I presented my lecture "Overlapping: A Paradigm of Parallel and Sequential Processing" (PPS). So, I was really very happy when Mr. Ryo Osaki from Ampere called me and told me that Dr. K. Furukawa will accept me for the visit at ICOT.

It is interesting to stress that I was officially the second Yugoslav who visited ICOT (the first one was professor Suad Alagic from Sarajevo). In ICOT, we discussed my overlapping principle which seemed to be similar (in this discussion) to the concept of guarded Horn clauses (dr. Furukawa's assumption). It became clear that I had to develop the overlapping principle into a more free and asynchronouos model in a way being independent from a parallel processor architecture. Parallel processing was the main subject of our discussion at ICOT. At last they demonstrated to me the so called Personal Serial Inference Machine (PSIM) functioning as a work station for the development of the fifth generation computer at ICOT.

Before living Japan, professor Eiichi Goto called me to the hotel. It was a pleasant meeting after our previous encounter 14 years ago when he explained his extraordinary motivation for his undertakings. This was a satisfactory end for my first visit to Japan.

Our engineering group that visited Japan (in fact, two of them - Andrej Bekeš and Drago Novak - lived in Japan and Marko Kovačević, Saš Hadži, Dušan Salehar, and the author were the visitors) began in some funny way to chat and discuss our own possibilities for some kind of the fifth generation computer. This funny orientation grew into a minimal form of theoretical possibilities. So, may be, for the first time the questions were stated: Can Iskra Delta develop, produce, and market a parallel machine? Are we able to organize the proper development undertaking? And thus, the funny idea rose into strategic thinking. The consequence of this thinking was our second trip to Japan in June 1986.

In our second trip to Japan, the most important conversation was held with professor Eiichi Goto. He was the main attraction of the Tokyo's symposium on computer architecture (Fig. 3), for he demonstrated his own basic computer technology (based on several of his patents) and the concept of new, ultra-fast computer architecture. He was the one who revealed the possibilities of developing architectural concepts standing outside of the common and approved professional and official orientations; introducing, for instance, magnetics, Josephson junction, antennas, etc.

In Tokyo we met our friends from Ampere (Fig. 1 and Fig. 2), and we had a very pleasant conversation with professor Goto at the Tokyo University (Fig. 4 and Fig. 5). Visits were arranged to Tsukuba University and to some other sites where parallel machines have been demonstrated. By our second expedition to Japan, the essential technological treads for IDC Parsys project conception had been identified.

References

(EGT) M. Heidegger: Early Greek Thinking. (The Dawn of Western Philosophy) Harper & Row. San Francisco (1984).

(TSZ) F. Nietzsche: Thus Spoke Zarathustra. A Book for Everyone and no One. Penguin Books (1985).

(PPS) A. P. Železnikar: Overlapping: A Paradigm of Parallel and Sequential Processing. Informatica 10 (1986). No. 2, 3-17.

(STL) A. P. Železnikar: From Sapporo to Tokyo Back to Ljubljana (in Slovene). Informatica 10 (1986), No. 2, 68-74.

(Will be continued)

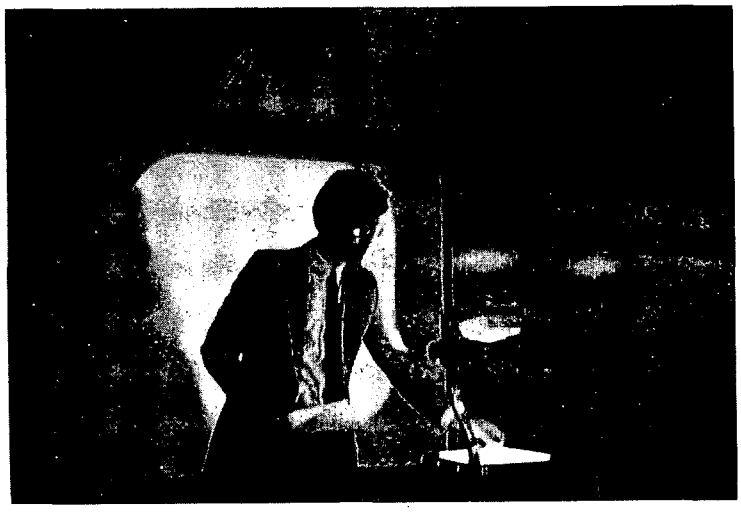


Figure 6. Mr. Petar Brajak, M. Sc., the chief designer of the IDC Parsys innovative parallel architecture, was the main lecturer of the MIT Seminar. His presentation awoke a life critical and constructive discussion, bringing to the surface new elements, influencing the speaker and audience.

Figure 7. Professor S. Prešern introduces the IDC Parsys Project at the Laboratory for Computer Science at MIT, giving notice to the innovative concept of the Parsys architecture and preparing the ground for Petar's performance.

Figure 8. Professor Robert H. Halstead (at the left), the host of the Seminar, was satisfied as well as the audience. Petar Brajak's smiling does not say that the lecture was interrupted by several counter-questions and that a creative effort was necessary to deliver the proper answers.

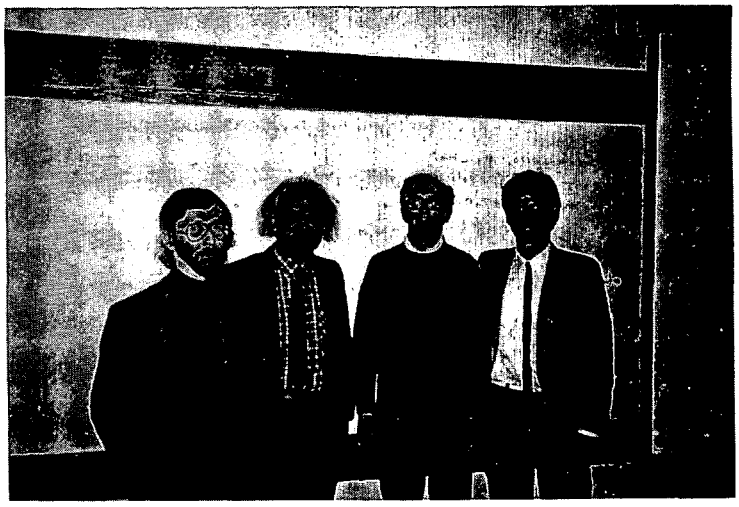


Figure 9. From left to right: professor S. Prešern, professor A. P. Železnikar, professor R. H. Halstead, and Mr. Petar Brajak (April 15, 1987, MIT).

LABORATORY FOR COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEMINAR

Date: Wednesday, April 15, 1987

Time: 2:45 - Refreshments, 3:00 - Talk

Place: 2nd Floor Lounge

Rationale And Concepts For The Parsys Parallel Processor Architecture

Petar Brajak

Iskradelta, Ljubljana, Yugoslavia

Adj. Prof. Sasa Presern

Jozef Stefan Institute, Ljubljana, Yugoslavia

Prof. A. P. Zeleznikar

Edvard Kardelj University, Ljubljana, Yugoslavia

Abstract

Parsys is a tightly coupled mimd parallel processor development project founded by Iskra Delta Computers and carried out by several institutions and faculty research groups in Yugoslavia. Specifically, the project is involved with:

- Development of prototype hardware and system software,
- Development of specific programming environments,
- Development of application software.

The architectural design of parsys is based upon already proven yet innovative concepts. The prototype machine will be a shared memory system consisting of 64 processors and 271 memory modules that are connected via a network of VLSI custom built ISMM (Intelligent Shared Memory Module) chips in a 6-cube configuration. Each ISMM supports fast routing and incorporates functions and logic that avoid or minimize degradations mostly encountered in the multiprocessor environments such as:

- N simultaneous accesses to the same memory location,
- N simultaneous accesses to the same memory module,
- N simultaneous synchronization requests,
- slow dynamic context switch,
- huge latency.

In this presentation we describe the concepts of the parsys project and show some of the ISMM functions. Primarily, we show how ISMMS enable two or more processors to read/write the same memory address in one memory cycle and how ISMMS maintain lists of waiting processes with a synchronizational mechanism that enables N processes to link up in only one memory cycle.

HOST: Professor R.H. Halstead

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

Figure 10 (at left). The poster announcing the Parsys Seminar at the MIT Laboratory for Computer Science.

ECE COMPUTER AREA SEMINAR

BY

PETER BRAJAK, SAŠA PREŠERN AND ANTON P. ŽELEZNIKAR

ISKRA DELTA, LJUBLJANA, YUGOSLAVIA

11:00 AM

FRIDAY

24 APRIL 1987

ENGR. 306

PARALLEL PROCESSOR ARCHITECTURES

1. PARSYS is a tightly coupled MIMD parallel processor development project founded by Iskra Delta Computers and carried out by several institutions and faculty research groups in Yugoslavia.

   The architectural design of PARSYS is based upon already proven yet innovative concepts. The prototype machine is a shared memory system consisting of 64 processors and 271 memory modules that are connected via network of VLSI custom build ISMM chips in an 6-cube configuration. Each ISMM supports fast routing and incorporates functions and logic that avoid or minimize degradation mostly encountered in the multiprocessor environments.

2. Trident is a 16 bit microcomputer with VME bus which offers the choice of three high performance CPU's based on microprocessors from Intel (80286), DEC (J11) and Motorola (68010). Trident's operating system is either XENIX (tm), MS/DOS (tm), UNIX (tm), RSX-11 (tm), CP/M 86 OS 9 (tm) and DELTA M (tm). Compatibility with IBM and DEC environment assures access to an enormous base of existing applications and development tools.

Figure 11 (above). The poster announcing the Parsys and Trident Seminar at the Electrical and Computer Engineering Department (ECE) of the University of Arizona in Tucson.

AVTORJEM ČASOPISA INFORMATICA

Zaradi možnega plasmaja časopisa Informatica v tujini naprošamo avtorje časopisa Informatica, da pišejo v prihodnje svoje članke, referate, eseje in novice po možnosti v angleščini. Povzetek v domačem jeziku z naslovom prispevka naj bo dodan na posebnem listu v formatu stolpca (47 znakov na vrstico).

AUTORIMA ČASOPISA INFORMATICA

Zbog mogućeg plasmana časopisa Informatica u inostanstvu, autori časopisa Informatica se umoljavaju da pišu u buduće svoje članke, referate, eseje i novosti po mogućnosti na engleskom. Abstrakt u domaćem jeziku sa naslovom članka neka bude dodat na posebnom listu i izpisan u formatu stupca (47 znakova na redak).

TO AUTHORS OF INFORMATICA

To achieve the possible exchange of Informatica with foreign journals, the authors of articles for Informatica are kindly requested to submit their articles, papers, essays, and news in English, when possible. Abstracts in a domestic language containing the title of an article should be submitted on a separate sheet and printed in the obvious column format (47 characters per line).