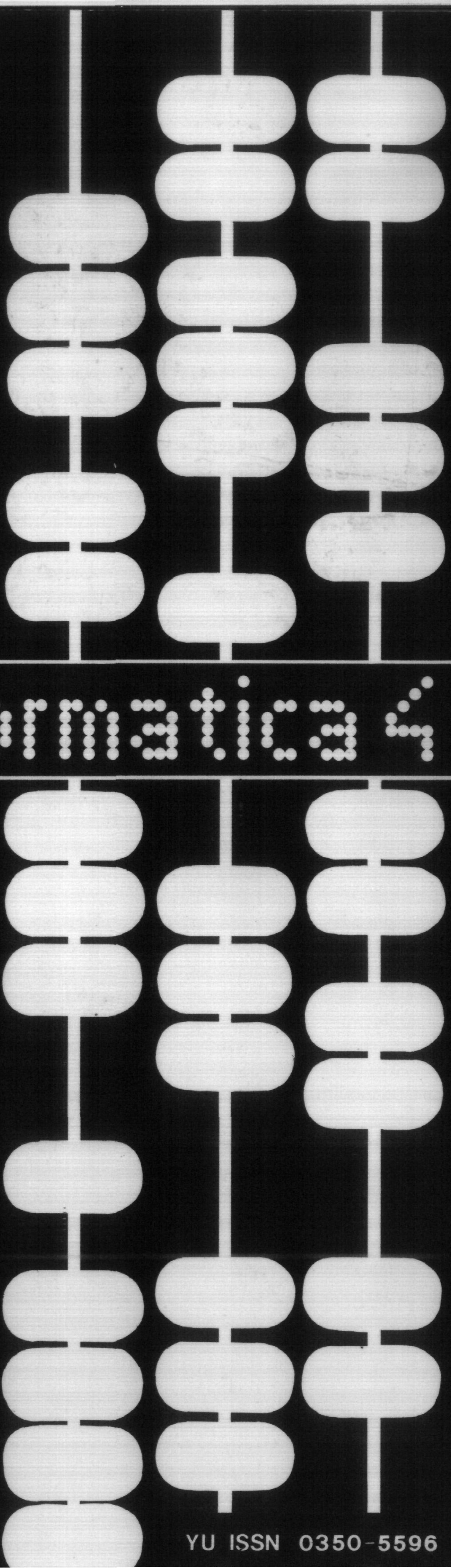


# 88 informatica 4

YU ISSN 0350-5596

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

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

Iskra Delta

# informatica

## JOURNAL OF COMPUTING AND INFORMATICS

YU ISSN 0350–5596

VOLUME 12, 1988 – No. 4

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

Headquarters:

**Informatica, Journal for Computing and Informatics, Iskra Delta Computers, Stegne 15C, 61000 Ljubljana, Yugoslavia**
**Phone: (+38 61) 57 45 54. Telex: 31366 yu delta**
**Fax: (+38 61) 32 88 87 and (+38 61) 55 32 61.**

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

**Opinions expressed in the contributions are not necessarily shared by the Editorial Board**

**Printed by: Tiskarna Kresija, Ljubljana**

C O N T E N T S

| | | |
|---|---|---|
| A.P.Zeleznikar | 3 | Informational Logic II |
| B. Dugonik | 21 | Temporal Logic - A Tool for System Modeling |
| I. Bruha | 25 | Compact Implementation of the Programming Language Prolog in the Environment of Mc Poplog |
| M. Stojanović<br>I. Stojmenović | 36 | On Deletion of Reflex Elements from Circular Doubly-Linked Lists |
| J. Barle<br>J. Grad | 39 | Computer Program for Determining an Optimum Solution in Long-Term Forest Exploitation Process |
| R. Andrejčič<br>B. Peček | 44 | Source Code Analysis |
| M. Ivanović<br>Z. Budimac | 49 | Translation of Grammar Rule EBNF Descriptions to BNF Ones |
| S. Bobek | 55 | Quo Vadis Informatization in Business Decision Making |
| B. Domanjko | 59 | UUCP Network |
| A.P.Zeleznikar | 63 | Information and Information System |
| | 69 | Forum Informationis |

# 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**

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

Uredništvo in uprava:

Časopis Informatika, Iskra Delta, Stegne 15C, 61000 Ljubljana, telefon (061) 574 554; teleks 31366 YU Delta; fax (061) 328 887 in (061) 553 261.

Letna naročnina za delovne organizacije znaša 24000 din, za zasebne naročnike 6000 din, za študente 2000 din; posamezna številka 8000 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 12, 1988 - ŠT. 4**

V S E B I N A

| | | |
|---|---|---|
| A.P.Železnikar | 3 | Informational Logic II |
| B. Dugonik | 21 | Temporal Logic - A Tool for System Modeling |
| I. Bruha | 25 | Compact Implementation of the Programming Language Prolog in the Environment of Mc Poplog |
| M. Stojanović<br>I. Stojmenović | 36 | On Deletion of Reflex Elements from Circular Doubly-Linked Lists |
| J. Barle<br>J. Grad | 39 | Computer Program for Determining an Optimum Solution in Long-Term Forest Exploitation Process |
| R. Andrejčič<br>B. Peček | 44 | Source Code Analysis |
| M. Ivanović<br>Z. Budimac | 49 | Prevodjenje pravila gramatike iz EBNF zapisa u BNF zapis |
| S. Bobek | 55 | Quo vadis informatizacija poslovnega odločanja |
| B. Domanjko | 59 | UUCP mreža |
| A.P.Železnikar | 63 | Informacija in informacijski sistem |
| | 69 | Forum informationis |

UDK 519.72

Anton. P. Železnikar
Iskra Delta, Ljubljana

This part of the essay deals with the formal informational logic, however only with a part of this logic, i.e. with the definitions of informational variables and operators which constitute the so-called list of primitive symbols. In the continuation of this essay formation rules, informational axioms, and informational transformation rules will be discussed in detail. The topic of this part of the essay is the following: Introduction into Formal Informational Logic and Basic Informational Variables and Operators. In the introductory section the syntax for labelling definitions, theorems, proofs, examples, and informational systems is exposed. Then, in the second section, various definitions of informational variables are introduced and three types of equivalence operators, used in the essay, are explained thoroughly. The next subsection is dedicated to general informational metaoperators, which are meant to be operational variables. These variables can be particularized and universalized into concrete operators (e.g., through a non-uniform substitution in a formula). Thereupon, operators concerning the informational cycle are determined. Within these, counter-informational operators and operators of embedding are emphasized. Operators of parallel Informing are discussed in a most exhaustive way. There are operators of general parallel Informing, parallel informational possibility and necessity, transfer of information among parallel processes, informational appearance and vanishing, informational choice, memorizing, forgetting, and renewing of information, informational interrupting and breakdown, and enriching of information. For better understanding, several examples of formulae including various operators are given. As it has already been mentioned, in the continuation of the essay, in Part Two, formation rules, informational axioms, and informational transformation rules will be included.

**Informacijska logika II. Drugi del: formalna informacijska logika.** Ta del spisa obravnava formalno informacijsko logiko, toda le del te logike, in sicer definicije informacijskih spremenljivk in operatorjev, ki oblikujejo t.i. seznam primitivnih simbolov. V nadaljevanju spisa bodo obravnavana podrobno še formacijska pravila, informacijski aksiomi in informacijska transformacijska pravila. Naslovna tema tega dela spisa je uvod v formalno informacijsko logiko in osnovne informacijske spremenljivke in operatorji. V uvodnem poglavju je opisana sintaksa označevanja definicij, teoremov, dokazov, primerov in informacijskih sistemov. V naslednjem poglavju se uvaja več definicij informacijskih spremenljivk in trije tipi ekvivalenčnih operatorjev s pojasnili njihove uporabe v tem spisu. Naslednje podpoglavje je namenjeno splošnim informacijskim (meta)operatorjem, ki se razumevajo kot operacijske spremenljivke, ki jih je mogoče konkretizirati (jim prirejati vrednosti) s t.i. partikularizacijo in univerzalizacijo (npr. z neuniformno substitucijo v informacijskih formulah). Potem se opredeljujejo operatorji informacijskega cikla. V okviru teh so posebej obravnavani protiinformacijski in vmestitveni operatorji. Operatorji paralelnega informiranja so najštevilnejši. To so operatorji splošnega paralelnega informiranja, paralelne informacijske možnosti in nujnosti, prenosa informacije med paralelnimi procesi, informacijskega nastanka in izginotja, informacijske izbire, pomnenja, pozabljanja in obnavljanja informacije, informacijskega prekinjanja in preloma in obogačevanja informacije. Z namenom boljšega razumevanja spisa je konstruiranih več primerov formul z različnimi operatorji. Kot je bilo že omenjeno, bodo v nadaljevanju tega spisa, v njegovem naslednjem delu opredeljena še formacijska (oblikovalna) pravila formul, informacijski aksiomi in informacijska transformacijska pravila formul.

## Part Two: Formal Informational Logic

### II.0. INTRODUCTION

*Logical thinking is an improvement of natural thinking because it trims the exuberance of natural thinking. Logical thinking is a deliberate attempt to restrain the excesses of natural thinking. This restraint is effected by selectively blocking natural flow pathways.*
*Logic is the management of NO.*

Edward de Bono [8] 224

In the first part of this essay a general and non-systematic (intuitive) view of informational logic and its possibilities were shown, dealing principally with the philosophy and conceptualism of informational formalization through introduction of a new semantics for basic logical operators. In this, second part of the essay, an *informationally axiomatic* approach will be given, constituting the so-called *symbolic system of informational logic.*

For the purpose of clarity of the new formalism, we will introduce several symbols for operators with already known, but also new semantics. Furthermore, we will use particular forms of marking and presenting definitions, theorems, proofs, and examples. Among these formalistic parts of the essay we shall put explanatory inserts for making formal achievements more picturesque and comprehensible. In a formal as well as in an intuitional manner, we shall develop and reveal a novel understanding of the arising informational formalism which, to our opinion, can constitute the possibility of today's and probably also of tomorrow's informational investigations and understanding.

To have a good perspicuity, we shall use the following syntax for *labelling* definitions (DF), theorems (TH), their proofs (PR), examples (EX), and informational systems (SY):

[Syntax of labelling]$^{\text{DF1}}$:

```
⟨label⟩    ::=
   [⟨characteristic_formula⟩]^⟨type⟩⟨number⟩:
⟨type⟩     ::=  DF|TH|PR|EX|SY
⟨number⟩   ::=  1|2|3|...
```

Definitions will begin by their label at the left of their first line and will end with the mark '■'. Theorems will begin by the label in the middle of their first line and will end with the mark '■'. Proofs, examples, and informational systems will begin by the label at the right of their first line and will end with the mark '■'. Generally, texts of definitions, theorems, proofs, examples, and informational systems will follow the general syntax

```
⟨label⟩
⟨text_of_the_type⟩                    '■'
```

This order of labelling will bring to the text the necessary systematics and uniformness. ■

### II. 1. SOME BASIC INFORMATIONAL VARIABLES AND OPERATORS

*The battle of reason is the struggle to break up the rigidity to which the understanding has reduced everything.*

G. W. F. Hegel [9] 53

In this section (II. 1.) we shall try to create a sufficiently broad basis of semantics of informational variables and operators, so that in the subsequent sections of this essay it would be possible to construct the axiomatic basis and the mechanisms of constructing informational formulae by means of proposed formation and transformation rules. The semantic basis of this section is relevant for it can substantially influence the possibilities of arising of informational logic. In this illumination we have to develop the semantics of informational variables and particularly of informational operators, considering the limits of the intuitive concept of information within the attainability of the human informational mind.

As in some recently discussed cases, we should like to distinguish clearly the variables concerning information, and those concerning Informing. Although information is in no way a passive informational component, in some expressions it will be put comprehensively in a passive state, and it will function as an operand. In this state information will be denoted by a small Greek letter. However, it is not to say that information will not influence other information or that it will not function as an implicit operator. Informing as an informational operator over information will be denoted by a capital Latin letter. However, it does not mean that Informing is not under the influence of other Informing or information, that is that it does not function as an (implicit) informational operand. Except of these two types (categories) of information (called information and Informing) we shall have some distinguished (logical) operators, considering the specifically investigational or informationally principal properties of information. These distinguished operators will constitute the logical or philosophical background of the presented informationally logical discourse.

#### II.1.0. Informational Variables

[Variables]$^{\text{DF1}}$:
Variables, presenting information in cases of operands and operators, will be denoted by distinguished small Greek or capital Latin letters and operational symbols. In cases of general information representing informational operands as variables, letters $\alpha$, $\beta$, ... , $\omega$ (non-indexed and indexed) will be used. Exceptionally also identifiers written in normal or Italic will be useful for representation of informational operands and

operators. Informational cases corresponding to particular forms and processes (e.g. counter-Informing, counter-information, embedding, information of embedding, metaphysics, etc.) will be marked by distinguished letters in indexed and non-indexed form. ■

**[Variables]$^{\mathrm{DF2}}$:**
Variables, representing information of a living being (cell, organism, etc.), will be, for instance, $\mu$ for metaphysics, $\sigma$ for sensory information, $\lambda$ for motor information, $\eta$ for emotional information, etc. Exceptionally also identifiers written in Italics, will be used. Further, $\pi$ will be used for marking the possibility, $\nu$ for the necessity and for the net of informational traces, $\lambda$ for marking informational lumps, $\tau$ for informational traces, etc. ■

**[Variables]$^{\mathrm{DF3}}$:**
Variables, presenting processes of Informing or Informings as operand variables, will be denoted by distinguished capital Latin letters. In the cases of general Informing, I (non-indexed and indexed) will be used. Exceptionally also identifiers written in Italic will be used. To Informing I corresponding counter-Informing and informational embedding will be denoted by C and E (non-indexed and indexed), respectively. ■

**[Variables]$^{\mathrm{DF4}}$:**
Variables and operators, presenting processes of Informing of a living being, will be denoted by M, S, L, Y, etc., for metaphysical Informing, sensory Informing, motor Informing, emotional Informing, etc., respectively. ■

**[Variables]$^{\mathrm{DF5}}$:**
The general variable representing any operator of informational logic will be the symbol $\models$ and other distinguished symbols declared as *informational operational variables*. These variables are representatives of generative sets of all (also the imaginative ones) possible informational operators. The values of these operational variables will be subscripted or superscripted operators of the type $\models$ or similar operator symbols or a semantically determined common symbols (e.g., $\wedge$, $\vee$, $\Rightarrow$, $\equiv$, $\in$, $\subset$, $\times$, $\cap$, $\cup$, $\circ$, $+$, $=$, etc.). It is to stress that informational logic introduces the institution of operators being informational variables in a similar sense as it introduces variables representing possible operands. In some cases, operational variables (markers for operators denoting variables) will be called *metaoperators*. The direction of the operation of an informational operator can be subscripted by $\rightarrow$ (from left to right) and by $\leftarrow$ (from right to left). ■

Although variables representing information and Informing will be passive and active or will have the roles of operands and operators, some especially distinguished information-logical or information-theoretical operators will be introduced into the "logical" discourse which follows. Some of these operators will be semantically broadened to satisfy also the requirements of the most general informational circumstances (informational phenomenology).

### II.1.1. Types of Equivalence Operators

Equivalence operators concern the opposite of the difference which is the most natural phenomenon of information. Informational equivalence (relations) must be considered to be purely theoretical, that is symbolical. Materialized, processible, or phenomenological cases of the so-called strict equivalence do not exist in nature; they can exist solely as informational phenomenon of abstraction, idealization, or, in general, as a being's metaphysical imagination.

**[Operators]$^{\mathrm{DF1}}$:**
Any concrete equivalence operator represents an operational value of the operational variable $\models$. Of course, we can also have an equivalence operational variable marked by $\models_{\equiv}$, for instance. In our discourse only three different types of equivalence operators will be used, namely,

| | | |
|---|---|---|
| '$=_{\mathrm{Df}}$' | for | *definitional* equivalence, |
| '$=$' | for | *symbolic* equivalence, and |
| '$\equiv$' | for | *logical* equivalence. ■ |

The definitional equivalence always introduces semantics of a new symbolism. The symbolic equivalence introduces (in fact, by definition) simply a new symbol, for instance, a shorter or a more adequate one. The logical equivalence concerns the classical logic (e.g. the truth and falsity) in informational well-formed formulae (iwffs).

### II.1.2. General Informational Metaoperators

*... we seek approved or verified definitions, the content of which is not assumed merely as given, but is seen and known to warrant itself, because warranted by the free self-evolution of thought.*

G. W. F. Hegel [9] 146

**[Operators]$^{\mathrm{DF2}}$:**
Two general metaoperators (informational, however operational variables) of Informing will be introduced. Let us have the following definition:

$$\models \vee \dashv \;=_{\mathrm{Df}}\; \begin{array}{l} \text{'informs'} \vee \\ \text{'inform'} \vee \\ \text{'is\_informed'} \vee \\ \text{'are\_informed'} \end{array}$$

In this case, $\models$ and $\dashv$ are understood to be unary, binary, or multiple prefix, postfix, or infix operators (as it will be presented later in several cases), which generally concern operand, operator or mixed information (for instance informational well-formed formulae within a logical system of information). In a general case (the so-called universal case) the operator $\models$ will incorporate also the meaning of the operator $\dashv$ and vice versa. Thus, the usage of the operator $\models$ will denote not only the meaning of both of them but also an arbitrary particular operational meaning. It is to say that operators $\models$ and $\Dashv$ will concern a general case of Informing as well as the so-called cyclical Informing or an integration of several informational cycles. In a similar way, they will represent also the so-called parallel informational cases of Informing. According to their application, needs, and the goals of an informational scenario, these metaoperators will be particularized (informationally specialized) by using subscripts and universalized (informationally generalized or variegated) by using superscripts. It is to say that operators $\models$ and $\Dashv$ are symmetrical with regard to the direction of operation, for instance, from the left to the right and vice versa, respectively. It would be quite sufficient to introduce only one of them and determine the second one as a particularization of the first one. ■

$[\text{Operators}]^{\text{EX1}}$:
The introduction of two operators $\models$ and $\Dashv$ instead of only one of them enables the expression of a parallel process in a single linear form. For instance, instead of two parallel processes $\alpha \models \beta$ and $\beta \Dashv \gamma$, it is possible to write simply $\alpha \models \beta \Dashv \gamma$. In this case $\beta$ is informed by $\alpha$ and simultaneously by $\gamma$. ■

$[\text{Operators}]^{\text{DF3}}$:
As mentioned in the previous definition, we can define the following two operators with their directionally different meanings:

$$\models^{\Dashv} \quad =_{\text{Df}} \quad \exists(\models, \Dashv).(\models \Delta \Dashv) \qquad \text{and}$$

$$\Dashv^{\models} \quad =_{\text{Df}} \quad \exists(\Dashv, \models).(\Dashv \Delta \models)$$

where these expressions are read as "there exist operators ... and ... such that (.) ...". $\Delta$ is the relation of dominance. In the first definition, operator $\Dashv$ is the so-called feed-back operator (or to $\models$ subordinated operator) and so is $\models$ in the second definition. In both cases, the relation of dominance $\Delta$ is not commutative. The feed-back operator is in fact functionally weaker than the main operator, so, it is put into the superscript position. We introduce the so-called operator (or relation) of the dominance of Informing $\Delta$, where for the first and for the second case

$$\models \Delta \Dashv \quad \text{and} \quad \Dashv \Delta \models$$

were valid, respectively. ■

$[\text{Operators}]^{\text{DF4}}$:

The operators $\models^{\Dashv}$ and $\Dashv^{\models}$ in the previous definition can be understood as universalizations of the operators $\models$ and $\Dashv$. In these cases it would be meaningful to introduce the superscript mark of universalization * in the following way:

$$\models^{*} =_{\text{Df}} \models^{\Dashv} \qquad \text{and} \qquad \Dashv^{*} =_{\text{Df}} \Dashv^{\models}$$

■

$[\text{Operators}]^{\text{DF5}}$:
In opposition to $[\text{Operators}]^{\text{DF2}}$ two general metaoperators (informational variables) of non-Informing can be introduced. Let us have the following definition:

$$\not\models \vee \not\Dashv \quad =_{\text{Df}} \quad \begin{array}{l} \text{'does\_not\_inform'} \vee \\ \text{'do\_not\_inform'} \vee \\ \text{'is\_not\_informed'} \vee \\ \text{'are\_not\_informed'} \end{array}$$

It is to understand that $\not\models$ and $\not\Dashv$ are unary, binary, or multiple prefix, postfix, or infix operators concerning a general, a cyclical, as well as a parallel informational case. It is to stress that a particularization of these operators does not mean a non-Informing for cases, which do not concern particularization. Thus, non-Informing really remains only particular. ■

$\not\models$ for instance has the meaning of a general non-Informing. The more universally the operator $\models$ is determined, the broader is the domain of non-Informing of the operator $\not\models$. A particularization of $\not\models$ localizes the non-Informing to a particular domain. In the case of $\not\models_{B}$ for instance, the domain of non-Informing is narrowed to the domain of belief (the subscript B). In classical logic, $\not\models_{T}$ can have the meaning of the non-true Informing, i.e. of the false Informing. In this respect $\not\models_{T}$ becomes logically equivalent to $\models_{F}$, etc.

$[\text{Operators}]^{\text{DF6}}$:
In a similar way as in $[\text{Operators}]^{\text{DF3}}$ four additional (universalized) operators of Informing can be defined, for instance:

$$\models^{\not\Dashv} \quad =_{\text{Df}} \quad \exists(\models, \not\Dashv).(\models \Delta \not\Dashv)$$

$$\not\models^{\Dashv} \quad =_{\text{Df}} \quad \exists(\not\models, \Dashv).(\not\models \Delta \Dashv)$$

$$\Dashv^{\not\models} \quad =_{\text{Df}} \quad \exists(\Dashv, \not\models).(\Dashv \Delta \not\models)$$

$$\not\Dashv^{\models} \quad =_{\text{Df}} \quad \exists(\not\Dashv, \models).(\not\Dashv \Delta \models)$$

The short form notation of these operators could be, for instance, $\models^{/*}$, $\not\models^{*}$, $\Dashv^{/*}$, and $\not\Dashv^{*}$, respectively. ■

As it can be understood from these four cases, a universalization means also absolutizing in the sense of Informing. In the last four cases universalization has in fact been unionization (for instance, in set-theoretical sense) of two informational operators, which becomes evident from the following definition:

$[\text{Operators}]^{\text{DF7}}$:
The universalization of two operators $\langle op_1\rangle$ and $\langle op_2\rangle$ can be achieved by defining the new operator in the following way:

$$\langle op_1\rangle^{\langle op_2\rangle} \quad =_{\text{Df}} \quad \langle op_1\rangle \vee_{\parallel} \langle op_2\rangle$$

where the defined universalized operator can be marked by

$$\langle op_1\rangle^{\langle s\rangle *} \quad \text{or} \quad \langle op_2\rangle^{\langle s\rangle *}$$

where $\langle s\rangle$ is the syntax variable of symbols and the syntax variables $\langle op_1\rangle$ and $\langle op_2\rangle$ are markers for generative classes (types) of informational operators. The disjunctive connective $\vee_{\parallel}$ emphasizes the parallelness of $\langle op_1\rangle$ and $\langle op_2\rangle$. Additionally, $\langle op_1\rangle \Delta \langle op_2\rangle$

can be defined. ■

**[Operators]$^{\mathrm{DF8}}$:**
Now, it becomes evident that also universalized operators of explicit non-Informing can be determined. According to the previous definitions of operators, the following two cases can be distinguished as consequently the non-Informing ones:

$$\not\models^{\not\Dashv} \;=_{\mathrm{Df}}\; \exists(\not\models, \not\Dashv).(\not\models \,\Delta\, \not\Dashv) \quad \text{and}$$

$$\not\Dashv^{\not\models} \;=_{\mathrm{Df}}\; u(\not\Dashv, \not\models).(\not\Dashv \,\Delta\, \not\models)$$

These operators can be denoted as $\not\models^{/*}$ and $\not\Dashv^{/*}$, respectively. ■

## II.1.3. Operators Concerning the Informational Cycle

Informational cycle needs to be additionally explained form the point of view of semantics. This cycle is under-stood to be an informational subprocess with some characteristic basic processes. In general two basic processes or characteristic cyclical sub-Informings can be distinguished: the so-called counter-Informing and embedding of the arisen information called counter-information. These two processes constitute the so-called Informing of the informational cycle. In this respect, similarly to the case of general Informing, two additional (generally cyclical) metaoperators can be introduced.

**[Operators]$^{\mathrm{DF9}}$:**
Two general operators (operational variables), explicitly concerning the informational cycle, can be introduced in the following way:

$\vdash \vee \dashv \;=_{\mathrm{Df}}$ 'informs_in_a_single_informational_cycle' $\vee$
'inform_in_a_single_informational_cycle' $\vee$
'is_informed_in_a_single_informational_cycle' $\vee$
'are_informed_in_a_single_informational_cycle'

Of course, the cyclical operators $\vdash$ and $\dashv$ can be understood to be subclasses of the general operators $\models$ and $\Dashv$, respectively. Also, the operator $\vdash$ can be universalized to incorporate the meaning of the operator $\dashv$, and vice versa. It is to stress that the operators $\vdash$ and $\dashv$ concern explicitly the so-called cyclic Informing. If these two operators are particularized, the particularization can concern only the so-called cyclical nature of Informing. Thus, each particularized or universalized operator of the type $\vdash$ or $\dashv$ concerns cyclical or intercyclical (cyclical) Informing. ■

Let us show and discuss some recursive definitions of cyclical operators and the meaning proceeding from these definitions.

**[Operators]$^{\mathrm{DF10}}$:**
Similarly to the [Operators]$^{\mathrm{DF3}}$, it is possible to determine in a recursive way two bi-directional (universalized) operators, namely:

$$\vdash^{\dashv} \;=_{\mathrm{Df}}\; \exists(\vdash, \dashv).(\vdash \,\Delta\, \dashv) \quad \text{and}$$

$$\dashv^{\vdash} \;=_{\mathrm{Df}}\; \exists(\dashv, \vdash).(\dashv \,\Delta\, \vdash)$$

where these expressions are read as "there exist two cyclical operators ... and ... such that (.) ...". In the first definition, operator $\dashv$ is the so-called cyclical feed-back operator (or to $\vdash$ subordinated cyclical operator) and so is $\vdash$ in the second definition. As $\vdash$ and $\dashv$ are already cyclical, the operators $\vdash^{\dashv}$ and $\dashv^{\vdash}$ in fact concern intercyclical processes (as cycles between two cyclical processes). However, cycles can include sub-cycles in a serial and parallel form, thus, cyclical (and non-cyclical) operators can appear without hesitation within informational cycles. Similarly to the definition [Operators]$^{\mathrm{DF4}}$, these two general intercyclical operators can be denoted in their universalized form in the following way:

$$\vdash^{*} \;=_{\mathrm{Df}}\; \vdash^{\dashv} \quad \text{and} \quad \dashv^{*} \;=_{\mathrm{Df}}\; \dashv^{\vdash}$$

■

**[Operators]$^{\mathrm{DF11}}$:**
In opposition to the [Operators]$^{\mathrm{DF10}}$, according to the [Operators]$^{\mathrm{DF5}}$, two cyclically general metaoperators (informational variables) of cyclical non-Informing can be introduced. So, let us have the following definition:

$\nvdash \vee \not\dashv \;=_{\mathrm{Df}}$ 'does_not_inform_cyclically' $\vee$
'do_not_inform_cyclically' $\vee$
'is_not_informed_cyclically' $\vee$
'are_not_informed_cyclically'

If Informing is not cyclical, it can still be the general, parallel, or non-cyclically particular one. In this respect, the cyclical non-Informing is an informational particularity too, concerning the concept of informational cycle. ■

The semantics of the informational cycle has been constituted through the introduction of two characteristic cyclical informational processes: counter-Informing and embedding of arisen counter-information. Within the cycle, the counter-Informing is the so-called progressive (informationally generative) process, in which counter-information arises. The embedding of arisen information is the so-called regressive (informationally connective) process, which is necessary to keep the arisen counter-information preserved for future Informing. The informational cycling becomes possible through the circulation of progressive (counter-informational) and to these processes following regressive (informationally embedding) processes. However, in this circular or cyclical process, counter-Informing and embedding are cyclical by themselves or they appear as (parallel) sub-cycles of the informational cycle. And this has to be kept in the concept of the informational cycle!

**[Operators]$^{\mathrm{DF12}}$:**
Eight particular, but at the same time general enough operators of the informational cycle are the operators concerning the cyclical counter-Informing and embedding. It is to understand that these operators have a metameaning and

have to be distinguished from concrete Informings, denoted by capital Latin letters. Particular metaoperators can be obtained by indexing (specializing) of the general meta-operators. Thus, the following four definitions for these operators can be proposed:

$\vdash_C \vee \dashv_C \;=_{Df}$ 'counter-informs_in_a_single_informational_cycle' $\vee$
'counter-inform_in_a_single_informational cycle' $\vee$
'is_counter-informed_in_a_single_informational_cycle' $\vee$
'are_counter-informed_in_a_single_informational_cycle'

$\nvdash_C \vee \not\dashv_C \;=_{Df}$ 'does_not_counter-inform_in_a_single_informational_cycle' $\vee$
'do_not_counter-inform_in_a_single_informational_cycle' $\vee$
'is_not_counter-informed_in_a_single_informational_cycle' $\vee$
'are_not_counter-informed_in_a_single_informational_cycle'

$\vdash_E \vee \dashv_E \;=_{Df}$ 'embeds_in_a_single_informational_cycle' $\vee$
'embed_in_a_single_informational_cycle' $\vee$
'is_embedded_in_a_single_informational_cycle' $\vee$
'are_embedded_in_a_single_informational_cycle'

$\nvdash_E \vee \not\dashv_E \;=_{Df}$ 'does_not_embed_in_a_single_informational_cycle' $\vee$
'do_not_embed_in_a_single_informational_cycle' $\vee$
'is_not_embedded_in_a_single_informational_cycle' $\vee$
'are_not_embedded_in_a_single_informational_cycle'

■

[Operators]$^{EX2}$:
Some possible definitions of the cyclical operators of Informing are the following:

$\vdash \;=_{Df}\; \vdash_C{}^{\dashv}E$, where

$$\vdash_C{}^{\dashv}E \;=_{Df}\; \exists(\vdash_C, \dashv_E)\cdot(\vdash_C \,\Delta\, \dashv_E)$$

$\dashv \;=_{Df}\; \dashv_C{}^{\vdash}E$, where

$$\dashv_C{}^{\vdash}E \;=_{Df}\; \exists(\dashv_C, \vdash_E)\cdot(\dashv_C \,\Delta\, \vdash_E)$$

In these definitions $\vdash_C$, $\vdash_E$, $\dashv_C$, and $\dashv_E$ are cyclical by themselves (by definition). Besides these cyclicities (the informational cycle and within it the cycles of counter-Informing and embedding) the cycles of counter-Informing and embedding can be understood as informational processes parallel to each other. ■

[Operator]$^{EX3}$:
Now we can put the following question: How can the operator $\vdash^{*}$ be informationally interpreted, keeping in memory that it was determined as $\vdash^{\dashv}$? It is possible to give the following definitional interpretation:

$$\vdash^{\dashv} \;=_{Df}\; \exists(\vdash_C, \dashv_E, \dashv_C, \vdash_E)\cdot ((\vdash_C \,\Delta\, \dashv_E) \,\Delta\, (\dashv_C \,\Delta\, \vdash_E))$$

This interpretation is a short-form expression of

$$\vdash^{\dashv} \;=_{Df}\; \exists(\vdash_C, \dashv_E, \dashv_C, \vdash_E)\cdot (((\exists(\vdash_C, \dashv_E)\cdot(\vdash_C \,\Delta\, \dashv_E)) \,\Delta\, (\exists(\dashv_C, \vdash_E)\cdot(\dashv_C \,\Delta\, \vdash_E)))$$

■

The operators $\vdash^{*}$ and $\dashv^{*}$ in [Operators]$^{DF10}$ are in some respect counter-cyclical, for $\vdash$ creates one cycle and $\dashv$ the other. These counter-cyclical processes become obvious in counter-directional counter-Informings $\vdash_C$ and $\dashv_C$ and in counter-directional embeddings $\vdash_E$ and $\dashv_E$. So far, we have explained to some extend only the operators $\vdash^{\dashv}$ and $\dashv^{\vdash}$. According to [Operators]$^{DF6}$, we have the following:

[Operators]$^{DF13}$:
Let us define the following four cases of complex cyclical operators and explain their meaning:

$$\vdash^{\not\dashv} \;=_{Df}\; \exists(\vdash, \not\dashv)\cdot(\vdash \,\Delta\, \not\dashv)$$
$$\nvdash^{\dashv} \;=_{Df}\; \exists(\nvdash, \dashv)\cdot(\nvdash \,\Delta\, \dashv)$$
$$\dashv^{\nvdash} \;=_{Df}\; \exists(\dashv, \nvdash)\cdot(\dashv \,\Delta\, \nvdash)$$
$$\not\dashv^{\vdash} \;=_{Df}\; \exists(\not\dashv, \vdash)\cdot(\not\dashv \,\Delta\, \vdash)$$

The short-form notation of these operators are, for instance, $\vdash^{/*}$, $\nvdash^{*}$, $\dashv^{/*}$, and $\not\dashv^{*}$, respectively. The operator $\vdash^{\not\dashv}$, in fact, is reduced to $\vdash$; for backward cyclical non-Informing can be simply excluded as an additional explanation. Irrespective of the relation of dominance, similarly happens to $\not\dashv^{\vdash}$. In this way, the operators $\nvdash^{\dashv}$ and $\dashv^{\nvdash}$ can be reduced into $\dashv$. ■

[Operators]$^{DF14}$:
Two complex cyclical operators of explicit non-Informing can be determined in the following way:

$$\nvdash^{\not\dashv} \;=_{Df}\; \exists(\nvdash, \not\dashv)\cdot(\nvdash \,\Delta\, \not\dashv) \quad \text{and}$$
$$\not\dashv^{\nvdash} \;=_{Df}\; \exists(\not\dashv, \nvdash)\cdot(\not\dashv \,\Delta\, \nvdash)$$

These operators can be denoted by $\nvdash^{/*}$ and $\not\dashv^{/*}$, respectively. Evidently these complex cyclical operators can be reduced to $\nvdash$ or $\not\dashv$. ■

### II.1.4. Operators Representing the General Type of Parallel Informing

#### *II.1.4.0. Introduction to Informational Parallelness*

Informational parallelness belongs to the most complex and semantically most pretentious problems. The state of the art of parallel Informing will depend substantially on the possibilities hidden in the meaning of the expressiveness of parallel informational operators. It is clear that we have to appropriate a sufficiently powerful set of parallel informational operators quite at the beginning of our logical discourse. This set of parallel operators must be diversely rich and has to cover imaginable possibilities of parallel informational connectedness, i.e. of parallel communication and informational interaction.

It became evident already in the framework of the general and cyclical Informing that we have been in many cases confronted with the appearance of the so-called parallel informational processes. If the processes are parallel, there should exist an informational connection among them, via communication and informational interaction. Parts of information are influencing other parts of information or themselves. There can be several levels and forms of information constituting the so-called informational interweave, for instance, on the level of operands and operators. First, we shall show a general (very formalistic) case of parallelness on the level of informational operators, not searching into details of the parallel structure. Then, we shall prepare the philosophy of parallel possibility and necessity of informational interweaving by introduction of particular operators. Next, operators of the informational transfer (communication) will be determined. And at the end of this section we shall deal with informational operators of appearance and vanishing, as well as of choice, memorizing, forgetting, renewing, interrupting, breakdown, and enriching of information.

#### *II.1.4.1. Operators of the General Parallel Informing*

In this subsection we shall in fact repeat the philosophy of general Informing, introducing operators of parallel Informing.

**[Operators]**$^{\mathrm{DF15}}$**:**
Two general metaoperators (informational variables) of parallel Informing will be introduced. Let there be the following definition:

$$\Vdash \vee \Dashv \;=_{\mathrm{Df}}\; \begin{array}{ll} \text{'informs\_in\_parallel'} & \vee \\ \text{'inform\_in\_parallel'} & \vee \\ \text{'is\_informed\_in\_parallel'} & \vee \\ \text{'are\_informed\_in\_parallel'} & \end{array}$$

As usual, $\Vdash$ and $\Dashv$ are understood to be unary, binary, or multiple prefix, postfix, or infix parallel informational operators. Here the parallelness concerns the appearance of at least two parallel operators. By introducing parallel operators we are in fact concerned with a system of parallel informational processes. The appearance of $\Vdash$ and $\Dashv$ means that there will exist at least one more operator of the type $\Vdash$ and $\Dashv$, respectively. The operators $\Vdash$ and $\Dashv$ point to the connectedness (parallelness) of the process possessing $\Vdash$ or $\Dashv$ with other informational processes, having operators of the parallel type. The parallel operators $\Vdash$ and $\Dashv$ could be particularized and universalized in the way we have described in [Operators]$^{\mathrm{DF2}}$. ■

**[Operators]**$^{\mathrm{DF16}}$**:**
It is evident that similarly to the previous cases of informational operators, parallel operators can be made complex too to obtain their bi-directional function:

$$\Vdash^{\Dashv} \;=_{\mathrm{Df}}\; \exists(\Vdash, \Dashv).(\Vdash \Delta \Dashv) \quad \text{and}$$

$$\Dashv^{\Vdash} \;=_{\mathrm{Df}}\; \exists(\Dashv, \Vdash).(\Dashv \Delta \Vdash)$$

The superscript operators in the last two expressions are the so-called feed-back parallel operators. This feed-back parallelness means that in an infix notation the right side process of the expression is additionally parallel connected to the left side (first expression), and vice versa (second expression). The relation of dominance $\Delta$ is understood in the usual manner, so that the left parallel operator dominates over the right one.

The operators $\Vdash^{\Dashv}$ and $\Dashv^{\Vdash}$ can be understood as universalizations of the operators $\Vdash$ and $\Dashv$, and marked by $\Vdash^{*}$ and $\Dashv^{*}$. If the universalization of the operators is comprehended as an operational complexion, the particularization always has the meaning of operational specialization. ■

**[Operators]**$^{\mathrm{DF17}}$**:**
In opposition to [Operators]$^{\mathrm{DF16}}$ two general parallel metaoperators (informational variables) of parallel non-Informing can be determined:

$$\nVdash \vee \not\Dashv \;=_{\mathrm{Df}}\; \begin{array}{ll} \text{'does\_not\_inform\_in\_parallel'} & \vee \\ \text{'do\_not\_inform\_in\_parallel'} & \vee \\ \text{'is\_not\_informed\_in\_parallel'} & \vee \\ \text{'are\_not\_informed\_in\_parallel'} & \end{array}$$

To inform in parallel has the meaning that the process in question informs more than one informational process concurrently. To be not informed in parallel means that at the most one process of Informing has the influence on the process in question. ■

**Operators]**$^{\mathrm{DF18}}$**:**
In a similar way as in [Operators]$^{\mathrm{DF16}}$ the four additional complex operators of parallel Informing can be defined, for instance:

$$\Vdash^{\not\Dashv} \;=_{\mathrm{Df}}\; \exists(\Vdash, \not\Dashv).(\Vdash \Delta \not\Dashv)$$

$$\nVdash^{\Dashv} \;=_{\mathrm{Df}}\; \exists(\nVdash, \Dashv).(\nVdash \Delta \Dashv)$$

$$\Dashv^{\nVdash} \;=_{\mathrm{Df}}\; \exists(\Dashv, \nVdash).(\Dashv \Delta \nVdash)$$

$$\not\Dashv^{\Vdash} \;=_{\mathrm{Df}}\; \exists(\not\Dashv, \Vdash).(\not\Dashv \Delta \Vdash)$$

The abbreviated notation of these operators can be $\Vdash^{/*}$, $\nVdash^{*}$, $\Dashv^{/*}$, and $\not\Dashv^{*}$, respectively. The philosophy of these operators can be quite complex regarding the parallel Informing and parallel non-Informing. It is worth mentioning that distinct parallel operators in the above compositions can be particularized in different ways, so that parallel Informing and parallel non-Informing may not be in the opposition at all. ■

**[Operators]$^{\mathbf{DF19}}$:**
Now it becomes evident that also universalized operators of explicit parallel non-Informing can be determined. The following two cases can be distinguished as consequently the non-parallel Informing ones:

$$\nVdash^{\not\Dashv} \quad =_{\mathrm{Df}} \quad \exists(\nVdash, \not\Dashv)\centerdot(\nVdash \;\Delta\; \not\Dashv) \qquad \text{and}$$

$$\not\Dashv^{\nVdash} \quad =_{\mathrm{Df}} \quad \exists(\not\Dashv, \nVdash)\centerdot(\not\Dashv \;\Delta\; \nVdash)$$

These operators can be denoted as $\nVdash^{/*}$ and $\not\Dashv^{/*}$, respectively. By their appearance these operators exclude the so-called general parallel Informing from informational processes they concern. ■

### *II.1.4.2. Operators of Parallel Informational Possibility and Necessity*

Information and Informing are informationally interwoven phenomena. In principle, the interweaving of information is circularly spontaneous and represents merely another form of information, called informational interweaving. This interweaving can be comprehended as informational connectedness or, more generally, as informational parallelness. The question is how this parallelness arises and what are its possibilities and necessities.

A spontaneous way of informational arising does not limit the arising of connectedness among various entities, cases, and parts of information and Informing of information. In static information (algorithms, today's data structures), the connectedness of static informational items is fixed, determined as informational necessity. In general, the interweaving of information can be seen as a possibility and necessity of informational parallelness.

**[Variables]$^{\mathbf{DF6}}$:**
Let us introduce two general markers, variables, and operators $\pi$ and $\nu$, denoting the notions of possibility and of necessity, respectively. First, $\pi$ and $\nu$ will be used as markers, for instance, marking specific informational operators concerning possibility and necessity. Thus, it is possible to introduce informational operators

$$\models_{\pi},\ \vdash_{\nu},\ \wedge_{\pi},\ \not\Dashv_{\nu},\ \pi_{\|},\ \nu_{\|},\ \text{etc.}$$

Further, it is possible to use $\pi$ and $\nu$ as variables, for instance in

$$\alpha(\pi),\ \pi(\beta),\ I(\nu),\ \pi(C),\ \nu(E),\ \pi,\ \nu,\ \text{etc.}$$

Informational operators, concerning primarily the possibility and necessity, are for example:

$$\pi,\ \nu,\ \pi_{\|},\ \nu_{\|},\ \Dashv_{\pi},\ \nVdash_{\nu},\ \text{etc.}$$ ■

**[Variable]$^{\mathbf{DF7}}$:**
Let $\alpha(\pi_{\|})$ mean that information $\alpha$ is parallel accessible (in an informational input and output way), so that information $\pi_{\|}$ is arising as the possibility of a parallel informational connectedness (communication and interaction) among various informational entities. In this expression $\pi_{\|}$ is a regular informational variable of information $\alpha$. It is to say that $\pi_{\|}$ is a regular or an inherent property (an element of informational axiomatic basis) of the arising information $\alpha$. ■

**[Variables]$^{\mathbf{DF8}}$:**
Let $\alpha(\nu_{\|})$ mean that information $\alpha$ has a necessary (concrete) parallel structure $\nu_{\|}$ (in an informational input and output way). Of course, by $\nu_{\|}$, in $\alpha(\nu_{\|})$ also the parallel structure of $\alpha$ is determined, so that $\alpha$ can be decomposed in parallel informing processes which communicate and interact with other informational processes. Besides parallel informational connectedness with other information, $\nu_{\|}$ denotes the so-called necessary parallel structure and organization of $\alpha$. ■

**[Variables]$^{\mathbf{EX1}}$:**
Information of possible and necessary *information of parallel connectedness* $\pi_{\|}$ *and* $\nu_{\|}$ arises within information $\alpha$ through particular Informings which can be marked by $\models_{\pi}$ and $\models_{\nu}$, for example. In general, $\pi_{\|}$ is the information of possible parallel connectedness. E.g., $\pi_{\|}(\alpha)$ concerns the possibility of parallel connectedness of $\alpha$. Thus,

$$\alpha \models_{\pi} \pi_{\|}(\alpha)$$

denotes the process of arising of information concerning the possibility of parallel connectedness of information $\alpha$. Further, $\pi_{\|}(\alpha, \beta)$ concerns the possible parallel connectivity between $\alpha$ and $\beta$. To some extent, $\pi_{\|}$ belongs to an informational entity. So, $\pi_{\|}(\alpha, \beta)$ can belong to $\alpha$, to $\beta$, or to both of them, if $\alpha$ and $\beta$ "cooperate" informationally. So $\pi_{\|}(\alpha, \beta)$ may be split into $\pi_{\|}(\alpha)$ and $\pi_{\|}(\beta)$, as the possible parallel connectivities of $\alpha$ and $\beta$, respectively.

Similarly, $\nu_{\|}$ is the information of necessary parallel connectedness. E.g., $\nu_{\|}(\gamma)$ concerns the necessity of parallel connectedness of $\gamma$. Thus,

$$\gamma \models_{\nu} \nu_{\|}(\gamma)$$

denotes the process of arising of information concerning the necessity of parallel connectedness of information $\gamma$. Further, $\nu_{\|}(\gamma, \delta)$ concerns the necessary parallel connectivity between $\gamma$ and $\delta$. To some extent, $\nu_{\|}$ belongs to an informational item. Thus, $\nu_{\|}$($\gamma$

, $\delta$), for instance, can belong to $\gamma$, to $\delta$, or to both of them, if $\gamma$ and $\delta$ "cooperate" informationally. It is therefore possible that $\nu_{\parallel}(\gamma, \delta)$ be split into $\nu_{\parallel}(\gamma)$ and $\eta_{\parallel}(\delta)$, as the necessary parallel connectivities of $\gamma$ and $\delta$, respectively. ■

*II.1.4.3. Operators of Transfer of Information among Parallel Processes*

Information of possibility and necessity of informational connectedness is the essential point of informational parallelness. Certainly, informational processes can exist in parallel without any mutual informational influence (relation, transfer, dependence). It means that informational realms of completely isolated processes are separated too.

In this section our intention is to introduce some operators of informational connectedness which realize the so-called proper parallelism of informational processes or briefly the parallel informational system. As we can understand, the notion of parallelness surpasses the so-called sequentialness, concurrence, and temporalness of a system and turns philosophically into the structuralness and the organizing principles of information and Informing of information.

[Operators]$^{\mathrm{DF20}}$:
If in an informational formula operators of the types $\Vdash$ and $\Dashv$ appear, it means that this formula possesses some informational operators of connectedness with other formulae. There are several possibilities to determine the so-called operators of informational communication. The aim of these operators is to transfer information from one informational process to the other. In this way, we can introduce the following simple operators of transfer:

$$\uparrow \;=_{\mathrm{Df}}\; \begin{array}{ll} \text{'carries\_off'} & \vee \\ \text{'carry\_off'} & \vee \\ \text{'is\_carried\_off'} & \vee \\ \text{'are\_carried\_off'} & \end{array}$$

In this definition $\uparrow$ is a regular informational operator of carrying-off. As we can understand, this operator can have an active and a passive mode. If $\alpha \uparrow$, information $\alpha$ carries off and if $\uparrow \alpha$, information $\alpha$ is carried off. Further, if $\alpha \uparrow \beta$, information $\alpha$ carries off information $\beta$. It is possible to define $\uparrow_{\rightarrow} \vee \uparrow_{\leftarrow}$ instead of $\uparrow$ in the last definition.

For the second operator of transfer we take:

$$\downarrow \;=_{\mathrm{Df}}\; \begin{array}{ll} \text{'brings'} & \vee \\ \text{'bring'} & \vee \\ \text{'is\_brought'} & \vee \\ \text{'are\_brought'} & \end{array}$$

In this definition $\downarrow$ is a regular informational operator of bringing, which can have an active or a passive mode. If $\alpha \downarrow$, information $\alpha$ brings and if $\downarrow \alpha$, information $\alpha$ is brought. Further, if $\alpha \downarrow \beta$, information $\alpha$ brings information $\beta$ or information $\beta$ is brought by information $\alpha$. It is possible to define $\downarrow_{\rightarrow} \vee \downarrow_{\leftarrow}$ instead of $\downarrow$ in the last definition. ■

There always exist pairs of particular (fixed) operators $\uparrow$ and $\downarrow$. Further, we can imagine how these operators can be particularized and universalized. We can use $\models_{\uparrow}$ and $\models_{\downarrow}$, for instance, but also $\uparrow_{C}$ and $\downarrow_{E}$ for marking operators of transfer within counter-Informing and embedding, respectively. Universalizations of these operators could be $\uparrow^{\downarrow}$ and $\downarrow^{\uparrow}$, for instance. For complexed operators of informational transfer special symbols can be introduced.

[Operators]$^{\mathrm{DF21}}$:
According to the definition [operators]$^{\mathrm{DF20}}$, it is possible to determine operators which block the transfer of information. Thus, we can define two informational "blockers" in the following way:

$$\uparrow_{/} \;=_{\mathrm{Df}}\; \begin{array}{ll} \text{'blocks\_to\_carry\_off'} & \vee \\ \text{'block\_to\_carry\_off'} & \vee \\ \text{'is\_blocked\_to\_be\_carried\_off'} & \vee \\ \text{'are\_blocked\_to\_be\_carried\_off'} & \end{array}$$

and

$$\downarrow_{/} \;=_{\mathrm{Df}}\; \begin{array}{ll} \text{'blocks\_to\_bring'} & \vee \\ \text{'block\_to\_bring'} & \vee \\ \text{'is\_blocked\_to\_be\_brought'} & \vee \\ \text{'are\_blocked\_to\_be\_brought'} & \end{array}$$

These blocking operators of informational transfer prohibit explicitly the transfer of information in question in those informational systems, which concern the security, for instance, of life information. Further, in the last definitions, it is possible to define $\uparrow_{/\rightarrow} \vee \uparrow_{/\leftarrow}$ and $\downarrow_{/\rightarrow} \vee \downarrow_{/\leftarrow}$ instead of $\uparrow_{/}$ and $\downarrow_{/}$, respectively. ■

[Operators]$^{\mathrm{EX4}}$:
Let it arise information which can be characterized as catastrophic. This sort of information which regularly arises in an informational process can cause catastrophic consequences when transferred to another process. Let this information be denoted by $\varepsilon_{\lightning}$.

Thus, let us have an initial parallel system

$$\alpha \Vdash \beta, \quad \gamma \Vdash \delta$$

where in the first process catastrophic information $\varepsilon_{\lightning}$ arises as a consequence of parallel Informing $\Vdash$. So,

$$\alpha \Vdash \beta \mathrel{\mathrm{L}} \varepsilon_{\lightning}, \quad \gamma \Vdash \delta$$

The operator L marks the appearance of information $\varepsilon_{\lightning}$ within information $\beta$. Now, let $\alpha$ block the carrying-off of information $\varepsilon_{\lightning}$ and let $\gamma$ block $\varepsilon_{\lightning}$ to be brought into the second process. Thus, there is

$$(\alpha \uparrow_{/} \varepsilon_{\lightning}) \Vdash \beta, \quad (\gamma \downarrow_{/} \varepsilon_{\lightning}) \Vdash \delta$$

This completes the example using operators of blocking informational transfer. ■

[Operators]$^{\mathrm{DF22}}$:
Now, it is possible to define more general operators of informational connectedness, namely the operators of inter-informational

Informing. These operators inform from the environment of a given information, e.g., from information $\alpha$, using a particular sub-information $\varepsilon$ within $\alpha$ on the way from an informational transmitter to possible informational receivers (sinks). The way between a transmitter and a receiver is performing as regular Informing, so that these general operators of informational connectedness have the function of Informing, similar to the functions of general operators. These operators can certainly be particularized to achieve the functions of operators $\uparrow$ and $\downarrow$. The aim of these operators is to conduct information in an informational system not only through carrying off and bringing of information, but also through regular Informing of this information on the way to informational receivers. In this sense these operators perform as regular operators of Informing. Let it be:

$\Uparrow \; =_{\mathrm{Df}}$ 'sends_and_informs' $\vee$
'send_and_inform' $\vee$
'is_sent_and_informed' $\vee$
'are_sent_and_informed'

and

$\Downarrow \; =_{\mathrm{Df}}$ 'receives_and_informs' $\vee$
'receive_and_inform' $\vee$
'is_received_and_informed' $\vee$
'are_received_and_informed'

These operators can be read in the following way:

| | | |
|---|---|---|
| $\alpha \Uparrow$ | means | 'α sends and informs' |
| $\alpha, \beta \Uparrow$ | means | 'α and β send and inform' |
| $\Uparrow \alpha$ | means | 'α is sent and informed' |
| $\Uparrow \alpha, \beta$ | means | 'α and β are sent and informed' |
| $\alpha \Uparrow \beta$ | means | 'α sends and informs β' or 'β is sent and informed by α' |
| $\alpha, \beta \Uparrow \gamma, \delta$ | means | 'α and β send and inform γ and δ' or 'γ and δ are sent and informed by α and β' |

and

| | | |
|---|---|---|
| $\alpha \Downarrow$ | means | 'α receives and informs' |
| $\alpha, \beta \Downarrow$ | means | 'α and β receive and inform' |
| $\Downarrow \alpha$ | means | 'α is received and informed' |
| $\Downarrow \alpha, \beta$ | means | 'α and β are received and informed' |
| $\alpha \Downarrow \beta$ | means | 'α receives and informs β' or 'β is received and informed by α' |
| $\alpha, \beta \Downarrow \gamma, \delta$ | means | 'α and β receive and inform γ and δ' or 'γ and δ are received and informed by α and β' |

This completes the explanation of the operators of inter-Informing $\Uparrow$ and $\Downarrow$. In the last two definitions of operators $\Uparrow$ and $\Downarrow$, it is possible to put $\Uparrow_{\rightarrow} \vee \Uparrow_{\leftarrow}$ and $\Downarrow_{\rightarrow} \vee \Downarrow_{\leftarrow}$ instead of $\Uparrow$ and $\Downarrow$, respectively. ■

[Operators]$^{\mathrm{DF23}}$:
We can certainly define two non-inter-informational operators of Informing; these operators can also be understood as general operators of non-Informing particularized in the described manner. Thus, we have the following definitions:

$\Uparrow_{/} \; =_{\mathrm{Df}}$ 'does_not_send_and_does_not_inform' $\vee$
'do_not_send_and_do_not_inform' $\vee$
'is_not_sent_and_is_not_informed' $\vee$
'are_not_sent_and_are_not_informed'

and

$\Downarrow_{/} \; =_{\mathrm{Df}}$ 'does_not_receive_and_does_not_inform' $\vee$
'do_not_receive_and_do_not_inform' $\vee$
'is_not_received_and_is_not_informed' $\vee$
'are_not_received_and_are_not_informed'

Also in this case, in the upper definitions, operators $\Uparrow_{/}$ and $\Downarrow_{/}$ can be replaced by disjunctions $\Uparrow_{/\rightarrow} \vee \Uparrow_{/\leftarrow}$ and $\Downarrow_{/\rightarrow} \vee \Downarrow_{/\leftarrow}$. ■

### *II.1.4.4. Operators of Informational Appearance and of Informational Vanishing*

In this subsection we shall enter into the most substantial discourse of Informing which concerns informational appearance and informational vanishing. Informational appearance is for instance the arising of counter-information and the arriving of sensory information to human cortices. Informational vanishing is for instance the dieing of a being's metaphysics or the ceasing of information because of certain changes in an outside phenomenon.

[Operators]$^{\mathrm{DF24}}$:
Let us have the following definition of the operators of informational appearance:

$\llcorner \vee \lrcorner \; =_{\mathrm{Df}}$ 'causes_the_appearance_of' $\vee$
'cause_the_appearance_of' $\vee$
'comes_into_existence' $\vee$
'come_into_existence'

Information $\alpha$, which informs, causes the appearance of counter-information $\omega$, or counter-information $\omega$ is coming into existence through the Informing $I$ of information $\alpha$. Thus,

$$\alpha \models_{I} \omega \;\equiv\; \alpha \llcorner \omega \quad \text{or}$$
$$\omega =\!\mid_{I} \alpha \;\equiv\; \omega \lrcorner \alpha$$

■

[Operators]$^{\mathrm{EX5}}$:
It would be very useful for later axiomatic construction that we become aware of the significance of the previous definition, which directly induces the following axiomatic

formula:

$$\alpha \models \;\Rightarrow\; \mathsf{L}\,\omega$$

This formula says the following: if information α informs, then counter-information ω is coming into existence. But it is evident that this axiom is only partial, for we can put a more complete one, such as:

$$((\alpha \models) \vee (\models \alpha)) \;\Rightarrow\; (\mathsf{L}\,\omega)$$

Further, we have some other axiomatic possibilities, for example:

$$\alpha \models \;\Rightarrow\; \alpha\,\mathsf{L}\,\omega \quad \text{and}$$
$$\alpha\,\mathsf{L}\,\omega \;\Rightarrow\; \alpha \models$$

which yields

$$\alpha \models \;\equiv\; \alpha\,\mathsf{L}\,\omega$$

etc. ■

**[Operators]$^{DF25}$:**
According to [Operators]$^{DF24}$, it is possible to define two operators of informational non-appearance, for instance:

$\mathsf{L}_/ \vee \lrcorner_/ \;=_{Df}$ 'does_not_cause_the_appearance_of' ∨
'do_not_cause_the_appearance_of' ∨
'does_not_come_into_existence' ∨
'do_not_come_into_existence'

In case of these operators, for instance, the outside information does not appear. It means that the informational environment is not delivering information which could, in the form of, let us say sensory information, arrive to the human cortices. Also, these operators represent Informing which lacks the substantial component of Informing, namely the coming of counter-Information into existence. In this second case we have to do with a technological system, which does not counter-inform (which does not have the component of informational arising), but can accept information coming from its environment and produce information in a traditional way, through its algorithmic performing. ■

**[Operators]$^{EX6}$:**
According to [Operators]$^{DF5}$ it is possible to construct several axiomatic formulae, as the consequences of [Operator]$^{DF25}$. Thus, we can have the following axiomatic presumption:

$$\alpha \not\models \;\Rightarrow\; \mathsf{L}_/\,\omega$$

This formula says the following: if information α does not inform, then counter-information ω is not coming into existence. However, it is obvious that this axiom expresses only a logical particularity, for one can put the following example:

$$((\alpha \not\models) \wedge (\not\models \alpha)) \;\Rightarrow\; (\mathsf{L}_/\,\omega)$$

Further, we can construct another axiomatic possibility, for instance:

$$((\alpha \not\models) \wedge (\not\models \alpha) \wedge (\alpha \not\dashv) \wedge (\not\dashv \alpha)) \;\Rightarrow\; ((\mathsf{L}_/\,\omega) \wedge (\lrcorner_/\,\omega))$$

Several similar examples can be constructed. ■

The operators of informational appearance indicate also the beginning of a new process, the arriving of new information, and, certainly, the coming of information into existence. In this way, the operators of appearance are specifically involved in the so-called arising of information or in Informing. The operators of informational vanishing are in a certain contradiction with the operators of appearing. They indicate the ending of an informational process, its abolishing. In this way, the operators of vanishing are specifically involved in the so-called ceasing or coming of information and its Informing to an end.

**[Operators]$^{DF26}$:**
It is possible to have the following definition of the operators of informational vanishing:

$\urcorner \vee \ulcorner \;=_{Df}$ 'causes_an_end_of' ∨
'cause_an_end_of' ∨
'comes_to_an_end' ∨
'come_to_an_end'

Information α, which informs, can cause an end of itself (comes to an end) or of information β. The notations of these facts are:

$$\alpha \urcorner \quad \text{or} \quad \ulcorner \alpha \qquad \text{and} \qquad \alpha \urcorner \beta \quad \text{or} \quad \beta \ulcorner \alpha$$

Informational items α, β, ... , γ, which inform, can cause an end of themselves (come to an end) or, for instance, of informational items ξ, η, ... , ζ. The notations of these facts are:

$$\alpha, \beta, \ldots, \gamma \urcorner \quad \text{or} \quad \ulcorner \alpha, \beta, \ldots, \gamma$$

and

$$\alpha, \beta, \ldots, \gamma \urcorner \xi, \eta, \ldots, \zeta \quad \text{or}$$
$$\xi, \eta, \ldots, \zeta \ulcorner \alpha, \beta, \ldots, \gamma$$ ■

**[Operators]$^{EX7}$:**
The previous definition can induce several consequences of axiomatic and theoretic nature. The most natural one would probably be

$$\alpha \models \;\Rightarrow_\pi\; \alpha \urcorner$$

where the operator $\Rightarrow_\pi$ has the meaning of "if '...', then it is possible that '...'". In this formula we have to do with possible implication or with implication of possibility. If information α informs, then it is possible that its Informing will be stopped (or will come to an end). Further,

$$\alpha \urcorner \;\Rightarrow\; \mathrm{I}_\alpha \urcorner$$

If α is coming to an end, then its Informing $\mathrm{I}_\alpha$ stops. In this case implication ⇒ is necessary. Next, it is also possible to express the stopping of a whole process, for example:

$$(\alpha, \beta, \ldots, \gamma \models \xi, \eta, \ldots, \zeta) \urcorner$$

In this case the particular process vanishes, while its informational components can still perform informationally. ■

[Operators]$^{DF27}$:
To the operators of vanishing $\urcorner$ and $\ulcorner$ it is possible to define two opposite operators, the operators of informational non-vanishing. It can be done in the following way:

$\urcorner_/ \lor \ulcorner_/ \;=_{Df}$ 'does_not_cause_an_end_of' $\lor$
'do_not_cause_an_end_of' $\lor$
'does_not_come_to_an_end' $\lor$
'do_not_come_to_an_end'

Through these two operators it is explicitly said that the stopping of an informational process cannot occur in this state of processing. So, the process is forced to continue its Informing. In a deductive theory, for instance, a well-formed formula, which is already a theorem, can be forced to be deductively developed (informed) to become a new theorem. Further; as long as sensors of a living being are intact (or are informing normally), sensory information $\sigma$ is appearing continuously (e.g. rhythmically), so $\alpha \urcorner_/$ ($\sigma$ is not coming to an end). ■

[Operators]$^{EX8}$:
By means of operators $\urcorner_/$ and $\ulcorner_/$ it is possible to construct some preaxiomatic formulae. Let us have the following axiomatic presumption:

$$\alpha \models \;\Rightarrow\; (\alpha \urcorner_/ \land \urcorner_/ \alpha)$$

This formula says the following: if information $\alpha$ informs, then information $\alpha$ does not cause an end of itself and does not come to an end. The last presumption can be broadened, of course:

$$((\alpha \models) \lor (\dashv \alpha) \;\Rightarrow\; ((\alpha \urcorner_/) \land (\urcorner_/ \alpha) \land (\ulcorner_/ \alpha) \land (\alpha \ulcorner_/))$$

In the last formula merely the case $\dashv \alpha$ was added to $\alpha \models$. Now, we can imagine the following implication considering all possible cases of Informing:

$$\alpha \models \beta \;\Rightarrow\; ((\alpha \urcorner_/ \alpha) \land (\alpha \urcorner_/ \beta) \land (\beta \urcorner_/ \alpha) \land (\beta \urcorner_/ \beta) \land ((\alpha \models \beta) \urcorner_/) \land (\urcorner_/ (\alpha \models \beta)))$$

This formula presumes that in the process of Informing $\alpha \models \beta$ none of its component, including the process of Informing itself, causes an end or is coming to an end. ■

### *II.1.4.5. Operators of Informational Choice*

Informational choice is the inherence of informational spontaneity. Informational choice concerns and presumes the existence and the informational arising of informational alternatives, among which in a given situation none, a single one, or some of them can be chosen for further consideration of Informing. The choice from a chaos of alternatives can take into consideration the necessity and possibility, a determined and an unforeseeable decision of choice, a simple (predicative) and a complex (informational) selection of informational items, etc.

A mechanism or operator of choice can use its own information of choice, a predicative criterion by which it decides on truth and falsity of a chosen element, an arising and not absolutely predictive mechanism of choice, etc. In our discourse of informational choice we shall use only two typical informational operators, namely, the simple choice ($\mid$) and the complex choice ($\Box$). Further, it would be possible to introduce operators of choice considering functions of possibility and necessity (for instance, $\mid_\pi$, $\Box_\pi$, $\mid_\nu$, and $\Box_\nu$).

[Operators]$^{DF28}$:
In this definition, the following simple informational operators of choice without and with predicative criterion will be determined: $\mid$, $\mid_\rightarrow$, $\mid_\leftarrow$, $\mid_\nu$, and $\mid_\pi$. Let be:

$\mid \;=_{Df}$ 'chooses_among_alternatives' $\lor$
'choose_among_alternatives' $\lor$
'is_in_the_process_of_choosing_as_possible_alternative_(by)' $\lor$
'are_in_the_process_of_choosing_as_possible_alternatives_(by)'

The operator $\mid$ is understood to be oriented from left to right, so that one can use the symbol $\mid_\rightarrow$, which unambiguously shows the direction of the operation or $\mid_\leftarrow$ for the opposite direction. Thus, we get the following meanings:

| | |
|---|---|
| $\alpha \mid$ | $\alpha$ chooses among alternatives (which are not specified yet); |
| $\alpha, \beta, \ldots, \gamma \mid$ | $\alpha, \beta, \ldots, \gamma$ choose among alternatives (which are not specified yet); |
| $\mid \alpha$ | $\alpha$ is in the process of choosing as possible alternative by not yet determined information of criteria; |
| $\mid \alpha, \beta, \ldots, \gamma$ | $\alpha, \beta, \ldots, \gamma$ are in the process of choosing as possible alternatives; |
| $\alpha \mid \beta, \gamma, \ldots, \delta$ | $\alpha$ chooses among listed alternatives $\beta, \gamma, \ldots, \delta$; |
| $\alpha, \beta, \ldots, \gamma \mid \xi, \eta, \ldots, \zeta$ | $\alpha, \beta, \ldots, \gamma$ choose among alternatives $\xi, \eta, \ldots, \zeta$ or $\xi, \eta, \ldots, \zeta$ are in the process of choosing as possible alternatives by criteria $\alpha, \beta, \ldots, \gamma$. |

The criteria of choosing can have the nature of possibility. In this case, the criteria of choice can be chosen. Thus, the operator $\mid_\pi$ has the function of choosing possible criteria (from a generative set) and by the chosen criterion to choose alternatives. In contrast to $\mid_\pi$, the operator $\mid_\nu$ has the fixed, necessary criteria already at a disposal and uses them in a deterministic way. ■

[Operator]$^{DF29}$:
Let us define the complex informational operator of choice in the forms of $\square$, $\square_{\rightarrow}$, $\square_{\leftarrow}$, $\square_{\vee}$, and $\square_{\pi}$. Let be:

$\square$ $=_{Df}$ 'chooses_among_alternatives_and_simultaneously_informs' ∨
'choose_among_alternatives_and_simultaneously_inform' ∨
'is_in_the_process_of_choosing_as_possible_alternative_(by_criteria_...)_and_is_simultaneously_informed' ∨
'are_in_the_process_of_choosing_as_possible_alternatives_(by_criteria_...)_and_are_simultaneously_informed'

The consequence of this definition is that the operator $\square$ informs its alternatives by its criteria, and also vice versa (if it is universalized), during the process of choosing. In this way, alternatives depend on the criteria and these depend on the alternatives through the Informing of the choice operator $\square$. It is usually understood that the operator $\square$ performs from left to right, where the criteria of choice are on the left and the alternatives on the right of the operator. In this case the notation $\square_{\rightarrow}$ can be used. The notation $\square_{\leftarrow}$ can be used in the case when positions of the criteria and of the alternatives are swapped. The criteria of choosing can have the nature of possibility, so that they can be chosen in an informationally operative sense. Thus $\square_{\pi}$ has the meaning of free choosing the criteria in the case of $\square_{\pi}\ \alpha$, for instance. In the case of $\square_{\vee}$, the set of criteria is fixed by the list of the so-called necessary choice-criterion information. ■

[Operator]$^{EX9}$:
Let us explain the meaning of some formulae, using the operators of the type $\square$. For instance:

$\alpha\ \square$ — $\alpha$ (as criterion) chooses among the alternatives (which are not specified yet) and informs;

$\alpha, \beta, \ldots, \gamma\ \square$
criteria $\alpha, \beta, \ldots, \gamma$ choose among the alternatives (which are not defined yet) and inform;

$\square\ \alpha$ — the alternative $\alpha$ is in the process of choosing as possible alternative by not yet determined criteria and is informed;

$\square\ \alpha, \beta, \ldots, \gamma$
alternatives $\alpha, \beta, \ldots, \gamma$ are in the process of choosing as possible alternatives and are informed;

$\alpha\ \square\ \xi, \eta, \ldots, \zeta$
the criterion $\alpha$ chooses among the listed alternatives $\xi, \eta, \ldots, \zeta$, where $\alpha$ informs and the alternatives are informed (and $\square$ can be universalized);

$\alpha, \beta, \ldots, \gamma\ \square\ \xi, \eta, \ldots, \zeta$
the criteria $\alpha, \beta, \ldots, \gamma$ choose among the listed alternatives $\xi, \eta, \ldots, \zeta$, where the alternatives are informed by the criteria and may also be vice versa. ■

### *II.1.4.6. Operators of Memorizing, Forgetting, and Renewing of Information*

*A memory is what is left behind when something happens and does not completely unhappen. This trace does not have to be in a special place, and it does not have to tell much about what has happened.*

Edward de Bono [8] 41

In this subsection we shall answer the question "What are memorizing, forgetting, and informational renewing as informational processes, illuminated through introduction of specific informational operators and their operands?" What happens in an informational realm which is being concerned with processes of memorizing, forgetting, and renewing of information? Obviously, these processes belong to the fundamental and substantial phenomenology of Informing, i.e. of the so-called informational arising.

[Variables]$^{DF9}$:
Let us first discuss the informational realm of entities, items, pieces, and units which constitute the domain of informational operands. Let us introduce the notion of *informational lump* $\lambda$, representing a free informational unit with a specific informational structure. Thus, one can imagine that information, which informs, in fact chooses, generates, changes, and maybe destroys lumps and connects them in a parallel manner as informational entities.

*A lump* $\lambda$ *informs as information.* Let the entities of a lump $\lambda$ be a construct

$$\lambda = \lambda^{-}, \lambda^{*}, \lambda^{+}$$

where

$\lambda^{-}$ is the so-called preinformational (initial or head) part,
$\lambda^{*}$ is the central (contents or meaning) part, and
$\lambda^{+}$ is the postinformational (final or tail) part

of a lump. Parts $\lambda^{-}$ and $\lambda^{+}$ connect a lump with other lumps, and vice versa. The preinformational connectedness means that the lump in question has arisen as a consequence (motivation) of Informing on already existing lumps. The postinformational connectedness of a lump means that this lump represents a kind of semantic basis (motivation), which contributes to the semantics of lumps to which it is connected. Certainly, the connectedness of lumps can be arbitrarily circular and arises spontaneously.

Lumps as informational variables and as given informational entities can be arbitrarily subscripted and particularized. A lump $\lambda$ informs in a parallel way and this fact can be expressed symbolically by the following system:

$$\lambda = \lambda^{-}, \lambda^{*}, \lambda^{+}$$
$$\lambda \Vdash \lambda$$

This system means that during the life of a lump $\lambda$, it participates in connected processes concerning informational lumps and that it informs also autonomously in itself, according to $\lambda \Vdash \lambda$ as a system of cooperating parallel processes

$$\lambda^{-} \Vdash \lambda^{-},\quad \lambda^{-} \Vdash \lambda^{*},\quad \lambda^{-} \Vdash \lambda^{+},$$
$$\lambda^{*} \Vdash \lambda^{-},\quad \lambda^{*} \Vdash \lambda^{*},\quad \lambda^{*} \Vdash \lambda^{+},$$
$$\lambda^{+} \Vdash \lambda^{-},\quad \lambda^{+} \Vdash \lambda^{*},\quad \lambda^{+} \Vdash \lambda^{+}$$

These nine parallel processes constitute the informational lump $\lambda$ as information. ■

**[Variables]**$^{\text{DF10}}$:
The next notion relevant for processes of memorizing and forgetting of information is the so-called informational trace. A lump can participate in several traces. A trace $\tau$ is an informationally connected set of lumps $\lambda_1, \lambda_2, \ldots, \lambda_n$, thus

$$\tau = \lambda_1, \lambda_2, \ldots, \lambda_n$$

The connectedness of lumps in a trace is conditioned by their head and tail parts, $\lambda_i^{-}$ and $\lambda_i^{+}$, respectively. Memorizing means to proceed through the $\lambda^{+}$-part of a lump to the $\lambda^{-}$-part of another (next) lump to which the previous lump is informationally connected. Such a proceeding is possible also in the opposite direction, from the $\lambda^{-}$-part of a lump to the $\lambda^{+}$-part of a previous lump. The connectedness of the tail and the head parts is bi-directional, so, in the processes of memorizing Informing can be performed in both directions. This structure of the trace is an abstract construction. There could be many head and tail parts in a trace for its backward and forward connections to other traces.

The trace as information informs in itself. This Informing can be described by the parallel system

$$\tau = \lambda_1, \lambda_2, \ldots, \lambda_n$$
$$\tau \Vdash \tau$$ ■

**[Variables]**$^{\text{DF11}}$:
A memorizing net is a parallel informational system of informational traces. Thus, for a net $\nu$ we have:

$$\nu = \tau_1, \tau_2, \ldots, \tau_m$$
$$\nu \Vdash \nu$$

Memorizing and forgetting is a unique process of informational appearance and vanishing in a memorizing net. Memorizing means the appearance of connective information with the arising of the contents which subject the connectedness. Forgetting means the vanishing of connective information with the changing (also arising) of the contents in informational lumps. In general, Informing of information is nothing more than appearing and vanishing of information within the connected informational entities. ■

In general, memorizing can be understood to have two components: memorizing as writing (impressing, imprinting) of information to a memory trace (engram information) and memorizing as reading (recalling, remembering) of impressed information from a memory trace. The corresponding informational operators of impressing and recalling will be subscripted by m_w (memory_writing) and m_r (memory_reading), respectively.

**[Operators]**$^{\text{DF30}}$:
Memorizing means informational mapping or Informing of information $\sigma$ into a trace $\tau$. Memorizing is a kind of embedding of information into another information, is an operation of informational impressing or writing and can be represented by the operator

$$\Vdash_{m\_w} \vee \dashv\!\vdash_{m\_w} \;=_{\mathrm{Df}}\; \begin{array}{ll} \text{'is\_impressed\_in'} & \vee \\ \text{'are\_impressed\_in'} & \vee \\ \text{'is\_an\_impression\_of'} & \vee \\ \text{'are\_impressions\_of'} & \end{array}$$

The informational operators $\Vdash_{m\_w}$ and $\dashv\!\vdash_{m\_w}$ represent the adequate parallel Informings concerning the impressing of one or several informational entities in a trace or traces. ■

**[Operators]**$^{\text{EX10}}$:
From the previous definition of the memory writing operator we have, for instance:

| | |
|---|---|
| $\sigma \Vdash_{m\_w}$ | information $\sigma$ is impressed in (a trace not yet identified); |
| $\sigma_1, \sigma_2, \ldots, \sigma_m \Vdash_{m\_w}$ | the informational entities $\sigma_1, \sigma_2, \ldots, \sigma_m$ are impressed in (traces not yet identified); |
| $\Vdash_{m\_w} \tau$ | the trace $\tau$ is an impression of (information not yet identified); |
| $\Vdash_{m\_w} \tau_1, \tau_2, \ldots, \tau_n$ | the informational traces $\tau_1, \tau_2, \ldots, \tau_n$ are impressions of (informational entities not yet identified); |
| $\sigma \Vdash_{m\_w} \tau$ | information $\sigma$ is impressed in the trace $\tau$ or the trace $\tau$ is an impression of information $\sigma$; |

$\sigma_1, \sigma_2, \ldots, \sigma_m \Vdash_{m\_w} \tau$

the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$ are impressed in the trace $\tau$ or the trace $\tau$ is an impression of informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$; in this case the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$ are condensed as impressions in the trace $\tau$; we can say that the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$ are condensible memorized in the trace $\tau$;

$\sigma \Vdash_{m\_w} \tau_1, \tau_2, \ldots, \tau_n$

information $\sigma$ is impressed in the traces $\tau_1$, $\tau_2$, ..., $\tau_n$ or the traces $\tau_1$, $\tau_2$, ..., $\tau_n$ are impressions of information $\sigma$; in this case information $\sigma$ is distributedly memorized in the informational traces $\tau_1$, $\tau_2$, ..., $\tau_n$;

$\sigma_1, \sigma_2, \ldots, \sigma_m \Vdash_{m\_w} \tau_1, \tau_2, \ldots, \tau_n$

the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$ are impressed in the traces $\tau_1$, $\tau_2$, ..., $\tau_n$; in this case, the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$ are distributed as impressions over the traces $\tau_1$, $\tau_2$, ..., $\tau_n$, irrespective of the values of m and n; we can say also that the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$ are distributively memorized in the informational traces $\tau_1$, $\tau_2$, ..., $\tau_n$.

This example completes the possibilities of using the memory writing informational operator $\Vdash_{m\_w}$. For the informational operator $\dashv\!\vert_{m\_w}$ we can show similar formulae. ■

Information which is memorized in informational traces can be recalled (reminded, remembered, recollected) from informational traces by means of particular informational operators. However, it is to stress that during memorizing of information in informational traces this information is subjected to the Informing of traces and to the Informing of arriving information which concerns information memorized in traces. It means that during the memorization of information in traces this is changed and is regularly arising within informational processes in traces. This changing and arising of information during its memorization in traces can be called the informational forgetting.

**[Operators]$^{DF31}$:**
Information, memorized in a trace or several traces, can be recalled (recollected) by means of particular informational operators. Recalling of information means its reading from memory traces, so the subscript m_r for operators' particularization is used. We have the following definition:

| | | | |
|---|---|---|---|
| $\Vdash_{m\_r} \vee \dashv\!\vert_{m\_r}$ | $=_{Df}$ | 'recalls' | $\vee$ |
| | | 'recall' | $\vee$ |
| | | 'is_recalled_from' | $\vee$ |
| | | 'are_recalled_from' | |

The informational operators $\Vdash_{m\_r}$ and $\dashv\!\vert_{m\_r}$ represent the adequate parallel Informings concerning the recalling (reading) of information impressed in memory traces. ■

**[Operators]$^{EX11}$:**
Let us look at the most general example of recalling information by means of the operator $\dashv\!\vert_{m\_r}$:

$\sigma_1, \sigma_2, \ldots, \sigma_m \dashv\!\vert_{m\_w} \tau_1, \tau_2, \ldots, \tau_n$

the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$ are recalled from the memory traces $\tau_1$, $\tau_2$, ..., $\tau_n$ or the memory traces $\tau_1$, $\tau_2$, ..., $\tau_n$ recall the informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$; in this case information impressed in the memory traces is delivered in the form of informational entities $\sigma_1$, $\sigma_2$, ..., $\sigma_m$. ■

Forgetting of impressed information can be understood as nothing else but Informing of memory traces by similar, outward information, and Informing in the sense of informational decay of memory traces by themselves. The consequence of such Informings is the changing and arising of originally impressed information.

**[Operators]$^{DF32}$:**
Let us define the following two informational operators of forgetting:

| | | | |
|---|---|---|---|
| $\Vdash_f \vee \dashv\!\vert_f$ | $=_{Df}$ | 'disintegrates_and_informs' | $\vee$ |
| | | 'disintegrate_and_inform' | $\vee$ |
| | | 'is_disintegrated_and_is_informed' | $\vee$ |
| | | 'are_disintegrated_and_are_informed' | |

Informational forgetting is conditioned by disintegration (decomposing, losing, forgetting gradually) and Informing (by new incoming information) of impressed information. If incoming information informs impressed information, this kind of Informing represents a form of impressing of incoming information. ■

**[Operators]$^{EX12}$:**
As we have defined, the process of forgetting is influenced by two components: the decaying of impressed information by itself and the memorizing (impressing) of incoming information. In this respect forgetting is a parallel informational process, for instance,

$\tau, \alpha \Vdash_f \tau$

This process can be decomposed into the system

$$\tau \Vdash_f \tau, \quad \alpha \Vdash_f \tau$$

or into a similar parallel informational system

$$\tau \Vdash_{dis} \tau, \quad \alpha \Vdash_{m\_w} \tau$$

where the operators of disintegration (decaying) and impressing are used. The informational variable $\tau$ represents a memory trace or a net of such traces. ∎

The notion we have to introduce next is the informational similarity. Memorizing and forgetting as informational processes hide informational similarity in their backgrounds. Similarity of information represents one of the basic relations, similarly to informational difference. To be similar means to be different to some extent and different means to be similar to some extent.

**[Operators]**$^{DF33}$:
Let us have the following definition:

$\therefore \;=_{Df}$ 'informs_similarly' $\vee$
'inform_similarly' $\vee$
'is_informed_similarly' $\vee$
'are_informed_similarly'

As usually, this operator (relation) can be bi-directional, for instance $\therefore_{\rightarrow}$ or $\therefore_{\leftarrow}$ for Informing from the left to the right or vice versa, respectively. The operator of similarity can be understood to be symmetric, so, for example,

$$\alpha \therefore \beta \quad \equiv \quad \beta \therefore \alpha$$

Similarity is a substantial condition of memorizing, forgetting, and renewing of information. ∎

**[Operators]**$^{DF34}$:
Renewing of information belongs to the so-called cyclical parallel Informing. Its general informational operators could be discussed in Section II.1.5. Renewing of information informs in cycles composed of simultaneous impressing, recalling, and forgetting. In this process new information is entering (informing) memory traces. Thus, the operator of renewing $\Vdash_{rnw}$ can be defined by the parallel informational system in the following way:

$$\sigma \Vdash_{rnw} \tau \;=_{Df}\; \sigma \Vdash_{\therefore} \tau, \quad \sigma \Vdash_{m\_w} \tau, \quad \tau \Vdash_{m\_r} \sigma, \quad \tau, \sigma \Vdash_f \tau$$

In this system $\therefore$ was replaced by $\Vdash_{\therefore}$. In a similar way it is possible to define the operator $\dashv\!\vert_{rnw}$. In the process of informational renewing $\tau$ has the intention to reconstruct some information which contributed to the formation of the trace $\tau$. In this process there can be additional influence from sensory information $\sigma$ which currently enters into the informational system in question. ∎

In this subsection we have not been dealing with operators of the types $\nVdash$ and $\not\dashv\!\vert$, particularizing them for impressing, recalling, forgetting, and renewing of information.

### *II.1.4.7. Operators of Informational Interrupt and Breakdown*

At a certain point of the process of Informing, it is possible to say that the current Informing of (original) information was interrupted through the so-called interrupting information and that after interrupting a new, the so-called breakpoint information takes control over the process of Informing within the original information. If the new information is typically counter-informational, that is opposite and unexpected in respect to the original, interrupted information, then the new Informing is called breakdown. A breakdown as information informs in the opposite, essentially new sense, however still considering the interrupted information. The interrupting information, which causes interrupting of the governing information, can be counter-informational or outside-informational.

In our discourse we have identified three informational entities and their Informings: the original (interrupted), the interrupting, and the breakdown information. In a process of Informing, the phenomenon of interrupting original information and then Informing by the so-called breakdown information is called the *informational turn*. This turn brings into the Informing of information a new informational circularity and spontaneity which regenerates and renews previous (original) information. The informational turn is one of the essential events in the context of counter-information and counter-Informing. However, the turn is not only a new circularity; through spontaneity it is also a new informational quality, a new kind of information. The condition for the informational turn is the appearance of an interrupt and a breakdown. Let us now define some operators of these types of Informing.

**[Operators]**$^{DF35}$:
Let us have the following definition for informational operators of interrupting:

$\Vdash_{ir} \vee \dashv\!\vert_{ir} \;=_{Df}$ 'interrupts' $\vee$
'interrupt' $\vee$
'is_interrupted' $\vee$
'are_interrupted'

Further, let the operators of informational breakdown be defined as follows:

$\Vdash_{bdw} \vee \parallel_{bdw} \;=_{Df}$ 'breaks_down' $\vee$
'break_down' $\vee$
'is_broken_down' $\vee$
'are_broken_down'

These operators constitute the possibility of examination of the so-called phenomenology of informational turn. ∎

**[Operators]**$^{EX13}$:
Let $\mu$ be the original, $\pi$ the interrupting, and $\beta$ the breakdown information. For the sake of this example, let $\mu$ be the so-called metaphysics, i.e. the informational entirety of a living being. Let us have the following

parallel informational system:

| | |
|---|---|
| $\mu \Vdash \mu$, | (a being's contemplation) |
| $\pi \Vdash_{ir} \mu$, | (interruption of contemplation) |
| $\beta \subset_{\varepsilon} (\mu, \pi) \Vdash_{bdw} \mu$, | ($\beta$ breaks down $\mu$) |
| $\beta, \mu \Vdash \beta, \mu$, | ($\beta$ and $\mu$ inform mutually) |
| $\beta \Vdash_{\Delta} \mu$ | (however, $\beta$ informs dominantly) |

In the third line of the above system $\subset_{\varepsilon}$ means that $\beta$ is informationally embedded ($\subset$) by $\varepsilon$ in $\mu$ and in $\pi$ (interrupting information). ■

*II.1.4.8. Operators of Informational Enriching*

The enriching of information does not mean merely the accumulation of various information but also, and above all, the informational connectedness of the existing information. Rich information is characteristically interwoven and informs in an informationally interactive way. A scientific discipline, for instance, can be understood as information which through its scientific arising enriches itself. The arising of informational connectedness means the arising of new informational relations, forming new connectedness, which delivers additional scientific understanding. In the process of informational enriching, informational quantity and informational quality arise. Under informational quality also the arising of a new meaning and diversity of meanings can be understood.

Consequently, the informational enriching concerns the arising of quality and quantity of information in parallel (concurrently, simultaneously, and in an interweaving form). The informational quantity alone may not necessarily contribute to an essential enriching of information. Parallel and parallel cyclical Informings maybe the most natural ways of informational enriching. It is of course possible to conceptualize various particular and universal operators of informational enriching, with the aim to study general and special methods and theories of informational enriching and its informational arising. Operators of enriching can cause the arising of measures and criteria of informational quantity and informational quality of information and informational processes.

## II.1.5. Operators of Parallel Cyclical Informing

In this section we shall join and develop the concepts exposed in Sections II.1.3 (Operators Concerning the Informational Cycle) and II.1.4 (Operators Representing the General Type of Parallel Informing). In this section, the philosophy of information which has been developed, comes to questions such as the following: Is parallel Informing among different informational cycles at all possible? Are informational cycles conceptualized in principle and in a concealed manner as time-dependent sequences or can this time-dogmatical, causal, or if-then-principled glimpse be surpassed informationally? Is it possible to view a sequence of informational cycles as a complex informational process, which hides informational parallelness among different informational cycles? Does this concept exceed the possibilities of informational processing in a living organism and in a technological machine because of the so-called time barrier?

In this section we shall not block our discourse and the development of a theory of parallel Informing by the use of any time-dogmatical barrier. Thus, we shall allow parallel Informing also among different informational cycles. We could say that time in this discourse will function as any other regular information, which is, in its nature, causal, consecutive, sequential, serial, etc. We shall not accept a concept of an informational cycle as time-dependent construction among cycles in an informational process, but as informational process in its entirety and its informational temporality.

$[\text{Operators}]^{DF36}$:
Two general operators (operational variables) for expressing the parallelness of the informational cycle can be introduced by the following definition:

$$\Vdash \vee \dashv\vdash \;=_{Df}\; \text{'informs\_in\_parallel\_in\_a\_single\_informational\_cycle'} \vee \text{'inform\_in\_parallel\_in\_a\_single\_informational\_cycle'} \vee \text{'is\_informed\_in\_parallel\_in\_a\_single\_informational\_cycle'} \vee \text{'are\_informed\_in\_parallel\_in\_a\_single\_informational\_cycle'}$$

This definition means that a process within an informational cycle informs into other processes of its own cycle and into the processes of other informational cycles, and is informed from the processes of its own and from other informational cycles. It is to understand that operators $\Vdash$ and $\dashv\vdash$ can be particularized and universalized according to the needs of their application. Further, it is to comprehend that similarly to Section II.1.4 (particularized) parallel informational operators of non-Informing, universalization, possibility, necessity, transfer, appearance, vanishing, choice, memorizing, forgetting, renewing, interrupting, breakdown, and enriching of information can be constructed. ■

$[\text{Operators}]^{DF37}$:
Let us define the usual universalization of the parallel cyclical operators in the previous definition as

$$\Vdash^{\dashv\vdash} \;=_{Df}\; \exists(\Vdash, \dashv\vdash).(\Vdash \Delta \dashv\vdash) \quad \text{and}$$

$$\dashv\vdash^{\Vdash} \;=_{Df}\; \exists(\dashv\vdash, \Vdash).(\dashv\vdash \Delta \Vdash)$$

In the first expression, the operator $\dashv\vdash$ is the so-called parallel, cyclical feed-back operator (which is subordinated to the main operator $\Vdash$); so is $\Vdash$ in the second expression (which is subordinated to the main operator $\dashv\vdash$). We can again introduce

$\Vdash^{*} =_{Df} \Vdash^{\dashV}$ and $\dashV^{*} =_{Df} \dashV^{\Vdash}$ ■

[Operators]$^{DF38}$:
As a contrary to the operators $\Vdash$ and $\dashV$, two parallel cyclical metaoperators (informational variables) of parallel cyclical non-Informing can be introduced by the following definition:

$\nVdash \vee \not\dashV =_{Df}$ 'does_not_inform_in_a_parallel-cyclical_way' $\vee$
'do_not_inform_in_a_parallel-cyclical_way' $\vee$
'is_not_informed_in_a_parallel-cyclical_way' $\vee$
'are_not_informed_in_a_parallel-cyclical_way'

If Informing is not simultaneously parallel and cyclical, it can still be generally parallel or generally cyclical. The operators $\nVdash$ and $\not\dashV$ can, for instance, hinder the parallelness in cyclical Informing, cyclicity in parallel Informing, or both of these types of Informing within general Informing. ■

[Operators]$^{DF39}$:
As in the definitions of general, cyclical, and parallel types of operators ([Operators]$^{DF6}$, [Operators]$^{DF13}$, and [Operators]$^{DF18}$, respectively, it is possible to define the following four cases of complex parallel cyclical operators and to explain their meaning:

$$\Vdash^{\not\dashV} =_{Df} \exists(\Vdash, \not\dashV).(\Vdash \Delta \not\dashV)$$
$$\nVdash^{\dashV} =_{Df} \exists(\nVdash, \dashV).(\nVdash \Delta \dashV)$$
$$\dashV^{\nVdash} =_{Df} \exists(\dashV, \nVdash).(\dashV \Delta \nVdash)$$
$$\not\dashV^{\Vdash} =_{Df} \exists(\not\dashV, \Vdash).(\not\dashV \Delta \Vdash)$$

The short-form notation of these operators can be $\Vdash^{/*}$, $\nVdash^{*}$, $\dashV^{/*}$, and $\not\dashV^{*}$, respectively. The operators in complex definitions can be particularized non-uniformly, for instance, as parallel cyclical counter-Informing and parallel cyclical embedding. The next step of universalization or complexation would be to complex the already complexed operators. ■

[Operators]$^{DF40}$:
Two complex informational operators of a strict parallel cyclical non-Informing can be defined in the following manner:

$$\nVdash^{\not\dashV} =_{Df} \exists(\nVdash, \not\dashV).(\nVdash \Delta \not\dashV) \quad \text{and}$$
$$\not\dashV^{\nVdash} =_{Df} \exists(\not\dashV, \nVdash).(\not\dashV \Delta \nVdash)$$

These operators can be marked by $\nVdash^{/*}$ and $\not\dashV^{/*}$, respectively. Obviously, these complex parallel cyclical operators can be reduced to $\nVdash$ and $\not\dashV$. ■

[Operators]$^{DF41}$:
According to [Operators]$^{DF12}$, parallel cyclical operators of counter-Informing and embedding can be defined. It is to understand that $\Vdash_C$ and $\Vdash_E$ are parallel within an informational cycle. We can list the following four definitions concerning the informational cycle:

$\Vdash_C \vee \dashV_C =_{Df}$ 'counter-informs_cyclically_in_parallel' $\vee$
'counter-inform_cyclically_in_parallel' $\vee$
'is_counter-informed_cyclically_in_parallel' $\vee$
'are_counter-informed_cyclically_in_parallel'

$\nVdash_C \vee \not\dashV_C =_{Df}$ 'does_not_counter-inform_cyclically_in_parallel' $\vee$
'do_not_counter-inform_cyclically_in_parallel' $\vee$
'is_not_counter-informed_cyclically_in_parallel' $\vee$
'are_not_counter-informed_cyclically_in_parallel'

$\Vdash_E \vee \dashV_E =_{Df}$ 'embeds_cyclically_in_parallel' $\vee$
'embed_cyclically_in_parallel' $\vee$
'is_embedded_cyclically_in_parallel' $\vee$
'are_embedded_cyclically_in_parallel'

$\nVdash_E \vee \not\dashV_E =_{Df}$ 'does_not_embed_cyclically_in_parallel' $\vee$
'do_not_embed_cyclically_in_parallel' $\vee$
'is_not_embedded_cyclically_in_parallel' $\vee$
'are_not_embedded_cyclically_in_parallel'

■

## References

[8] E. de Bono: *The Mechanisms of Mind*. Penguin Books, Harmondsworth, Middlesex, England (Reprinted 1977).

[9] G.W.F. Hegel: *The Science of Logic*. In Hegel's Logic, being Part One of the Philosophical Sciences (1830), translated by W. Wallace. Oxford, At the Clarendon Press, reprinted 1985.

# TEMPORAL LOGIC - A TOOL FOR SYSTEM MODELING

INFORMATICA 4/1988

UDK 519.711.3

Bogdan Dugonik
Technical Faculty Maribor

Abstract

This paper presents a model for processes, that communicate by message-passing, and the use of temporal logic for their description. The specification of process determines behaviour on its external ports, whence its internal behaviour is not significant. Specifications based on the model are compositional, that is, we get external behaviour of a system from specifications of its components. In the introduction a process recorded with traces is presented. A model is shown that represents a process specification by use of temporal logic. Temporal logic operators, examples of temporal assertions and some specifications for primitive and composed processes are given.

Povzetek

V članku je prikazan model za procese in mreže procesov, ki komunicirajo izključno s prehajanjem sporočil, in uporaba časovne logike za njihov opis. Specifikacija procesa določa obnašanje na njegovih zunanjih priključkih, dogajanje v njegovi notranjosti pa nas ne zanima. Opisi procesov so komponibilni, kar pomeni, da zunanje obnašanje mreže dobimo iz specifikacij njenih komponent. Pri tem prikrijemo njeno notranjo strukturo. V uvodu predstavimo zapis procesa s sledmi. Prikažemo tudi specifikacijo procesa in mreže procesov z uporabo časovne logike. Opisani so operatorji časovne logike, podanih je nekaj primerov časovnih izjav ter specifikacij enostavnih in sestavljenih procesov.

## Introduction

This paper treats processes that communicate exclusively through message passing. Messages are transmitted over input and output ports. Let us observe a process on its external ports for a time, up to some moment. We get a sequence of events. The sequence gives us a trace. The trace is an information about behaviour of the process up to that moment. Traces are appropriate to describe only terminating processes. The model of a system is modular, it is composed of some primitive processes. In a system output ports of processes are linked to inputs of another processes. Some input and output ports are not linked. Nonlinked ports are external ports and linked ports are internal ports.

A process is determined by a set of possible input-output behaviours . The behaviour on a set of input and output ports is an infinite sequence of observations :

$$\sigma = (t_0, In_0, Out_0, Rd_0), (t_1, In_1, Out_1, Rd_1), \ldots$$

t is a trace of events on input ports (input events) and events on output ports (output events).
Events represent communication between processes. Communication can be synchronous or asynchronous. If communication is synchronous, a process cannot send a message until the receiving process is ready to read the message on his input ports. If communication is asynchronous, a process can send message on his output ports as soon as it is ready without having to wait for the receiving process. In case of synchronous communication, input events present the data read by a process, and in case of asynchronous communication, they present the data that have appeared on input ports. Output events in both ways of communication present messages sent by a process at its output ports. An event is recorded in the trace as a pair (x,k), where x is a datum and k is the port name where the event has appeared. In and Out are boolean functions. If they are true for input - output ports, then we say the process is ready to

communicate. If logical values of some input port functions are true, then we say the process is ready to read on those ports. If logical values of some output port functions are true, then we say the process is ready to send on those output ports . Rd gives us the number of messages read up to the observed moment for each input port . It is clear that this number cannot be greater than the number of messages which have appeared on an input port. We could describe the process by writing down all possible sequences $\sigma$, which is vague and for more complicated cases practically impossible. Because of that , we apply a language which enables us to express all possible behaviours of a process.

## Temporal logic:

Natural languages are very expressive, but not precise, whereas formal languages are very precise but not so expressive. Formal languages have strictly defined semantics and syntax. It is impossible to say everything in them. But what you can say is unambiguous.

Temporal logic is a formal language. It allows us to make a temporal description of a process and also formal dealing of it. We present a model of temporal logic with infinite sequence of states

$$\sigma = s_0, s_1, s_2, \ldots$$

Assertions in temporal logic have a temporal meaning because they include some temporal operators. Language employs a set of basic symbols: individual variables and constants, function and predicate symbols. These are divided into two subsets: a set of local symbols and a set of global symbols.
Global symbols keep their values unchanged from state to state. Local symbols can vary their values from one state to another. We use boolean operators, classical operators and temporal operators. Existence quantification and universal quantification are also included in the system.

## The operators of temporal logic:

By w we denote a term or a formula that may contain some temporal operators.

- temporal operator "**always**":

  $\Box w$ means: w is true in this moment and will be true forever.

- temporal operator "**eventually**":

  $\Diamond w$ means: w will be true at least in one of following moments.

- temporal operator "**until**":

  $w_1 \mathcal{U} w_2$ means : $w_1$ is continually true until $w_2$ becomes true, and $w_2$ does indeed become true.

- temporal operator "**unless**":

  $w_1 \mathcal{W} w_2$ means: $w_1$ is continually true or there is a moment, when $w_2$ is true and up to the moment $w_1$ is continually true.

- temporal operator "**next**":

  $\circ$ means: w will be true in next moment.

## Temporal assertions:

Formulas given below are examples of some temporal assertions. They are compounded by using temporal operators mentioned above. We will write $\wedge$ for conjunction, $\neg$ for negation, and $\Rightarrow$ for implication.

| | |
|---|---|
| $\Box(u \Rightarrow \Diamond v)$ | - whenever u is true, it will eventually be followed by v. |
| $\Diamond \Box w$ | - eventually w will become true and it will remain thrue forever. |
| $\Box \Diamond w$ | - every instant is followed by some later instant, when w is true, so w is true indefinitely often. |
| $\Diamond v \Rightarrow ((\neg v) \mathcal{U} u)$ | - if v ever happens, its first occurrence is preceded by, or coincides with u. |

We choose some fixed reference moment named reference moment when interpreting a temporal assertion. It tells us, what holds at the reference moment, i.e. now, and what will happen in the future of it. It says nothing about what happened before. Instead of $\sigma = s_0, s_1, s_2, \ldots$ let us write $\sigma = \sigma_0, \sigma_1, \sigma_2, \ldots$ where $\sigma_i$ is an observation in behaviour $\sigma$ . An observation can be treated as a state. Usually, a state of a process reflects current values of variables of the process. Hence, an observation $\sigma_i$ is a state, which involves the history of process up to a current moment in its trace $t_i$. Specification of a process signifies how this process behaves. For this model we write process specifications in the form of temporal assertions.
The specification of the system P has the form <P> R , where R is an assertion, written in temporal logic. Specification <P> R is read: each behaviour of system P satisfies assertion R. <P> R is external specification, if R specifies behaviour on P's external ports. System specification is a conjunction of component assertions. The external specification is obtained by using a proof system.
The proof system for specifications includes rules for temporal logic, rules regarding the description of the domain of values, rules for designing a system, a set of system components with their precise specifications, and axioms which define the behaviour of the system. Specification defines two kinds of process properties: safety and liveness. Informally, safety specification asserts, what the process may do, and liveness specification asserts, what it must do.We will give some examples of specifications.

First, we show how certain ports of a process can be disabled, discarded from communication by using negation operator $\neg$ with In or Out function.
In the specification below input port $a_1$ and output port $b_1$ are disabled. Input a1 can not read the data, and b1 can not send the data.

$$< C > \quad \Box \neg \mathrm{In}(a_1) \;\wedge\; \Box \neg \mathrm{Out}(b_1)$$

Now, let us assume that communication is asynchronous. We will write external specifications of processes. That is, we will specify a process only by means of lengths of ports' traces, without using In, Out and Rd. By $|b|$ we denote the length of trace on port b. $b \sqsubseteq c$ means, that sequence b is a prefix of sequence c. First example below (fig.1.) gives a process with input port k and output port l. The process reads six values on input k and writes them on output l leaving out the first two values.

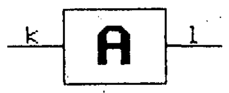


fig.1. Process with input port k and output port l.

$$\langle A \rangle\ \Box\, l \sqsubseteq [\ k(2),k(3),k(4),k(5)\ ]$$
$$\wedge\ (\ |l| = 0\ \mathcal{U}\ |k| \geq 3)$$
$$\wedge\ (\Diamond\ |k| = 3 \Rightarrow \Diamond\ |l| = 1)$$
$$\wedge\ (\Diamond\ |k| = 4 \Rightarrow \Diamond\ |l| = 2)$$
$$\wedge\ (\Diamond\ |k| = 5 \Rightarrow \Diamond\ |l| = 3)$$
$$\wedge\ (\Diamond\ |k| \geq 6 \Rightarrow \Diamond\ |l| = 4)$$

The first line is safety specification. It indicates that at most four values may occur on the output port, which will be read on input port k. The second conjunct indicates that the length of the output trace is zero, unless the length of input trace is three. The third conjunct and those following require the length of the output trace to be increased by one as soon the length of the input trace is increased.

Second example (fig. 2) is a process with two inputs and one output. It reads one value on j and one value on k. Then it writes first the value on the output l from input k, and later the value from input j.

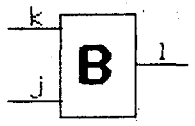


Fig.2. A process named B with two inputs and one output

$$\langle B \rangle\ \Box\, l \sqsubseteq [\ k(0),j(0)\ ]$$
$$\wedge\ (\ \Diamond\ |k| = 1 \Rightarrow \Diamond\ |l| = 1\ )$$
$$\wedge\ (\ \Diamond(\ |j| = 1 \wedge |k| \geq 1) \Rightarrow \Box\ |l| = 2)$$

Finally, we present an asynchronous network P of component processes A,B and C. Components A and B have one input and one output, component C two inputs and one output. Components are linked as shown in figure 3. Process A reads one value on input a and writes it twice on output c. Process B reads one value on input b and writes it once on output d. Process C reads values on its inputs, then writes them on output e as follows: first a value from input d, then both values from input c.

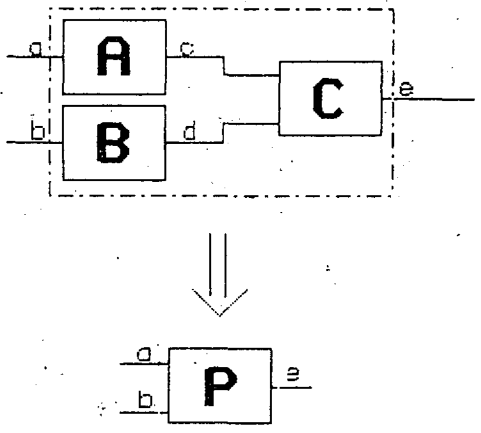


Fig.3. Process P with visible and hidden internal structure.

First, specifications for each components are given:

$$\langle A \rangle\ \ \Box\, c \sqsubseteq [\ a(0),a(0)\ ]$$
$$\wedge\ (\ \Diamond\ |a| \geq 1 \Rightarrow \Diamond\, \Box\ |c| = 2)$$

$$\langle B \rangle\ \ \Box\, d \sqsubseteq [\ b(0)\ ]$$
$$\wedge\ (\ \Diamond\ |b| = 1 \Rightarrow \Diamond\, \Box\ |d| = 1)$$

$$\langle C \rangle\ \ \Box\, e \sqsubseteq [\ d(0),c(0),c(1)\ ]$$
$$\wedge\ (\ \Diamond\ |d| \geq 1 \Rightarrow \Diamond\ |e| = 1)$$
$$\wedge\ (\ \Diamond\ (\ |c| = 1 \wedge |d| \geq 1) \Rightarrow (\ \Diamond\ |e| = 2)$$
$$\wedge\ (\ \Diamond\ (\ |c| = 2 \wedge |d| \geq 1) \Rightarrow (\ \Diamond\ |e| = 3)$$

External specification of P is obtained from conjunction of component specifications by using proof rules.

$$\langle P \rangle\ \ \Box\, e \sqsubseteq [\ b(0),\ a(0),\ a(0)\ ]$$
$$\wedge\ (\ \Diamond\ |b| \geq 1 \Rightarrow \Diamond\ |e| = 1\ )$$
$$\wedge\ (\ \Diamond\ (\ |a| \geq 1 \wedge |b| \geq 1) \Rightarrow \Diamond\, \Box\ |e| = 3\ )$$

It tells us how the system behaves on its external ports.

## Conclusion

We presented a model for processes which uses temporal logic as one of existing formal languages for specifying them. The benefit of temporal logic is its ability of describing temporal behaviour of processes. Specifications can be short and so expressive, that they need no extra comments. Troubles arise, if we want to set the exact time, because in the model we can express ourselves only qualitatively about the time of events.

## References:

[1] Dugonik B., Modeliranje procesov, diplomsko delo, Tehniška fakulteta v Mariboru, 1987

[2] Hoare C.A.R., Communicating Sequential Processes , Prentice - Hall International, Ltd., London 1985.

[3] Kapus T., časovna logika in mreže procesov, diplomsko delo, Tehniška fakulteta v Mariboru, 1987

[4] Lamport L., What Good is Temporal Logic ?, Information Processing (1983),pp. 657-668

[5] Manna Z.,Pnueli A., Verification of Concurrent Programs, Part I: The temporal Framework; Tehnical Report STAN-SC-81-836, Standford University, June 1981.

[6] Nguyen V., Demers A., Gries D., Owicki S., A model and temporal proof system for networks of processes, Distributed Computing (1986), 1, pp. 7-25.

# COMPACT IMPLEMENTATION OF THE PROGRAMMING LANGUAGE PROLOG IN THE ENVIRONMENT OF Mc POPLOG

INFORMATICA 4/1988

**Ivan Bruha**
**Mc Master University**
**Department of Computer Science and Systems**
**Hamilton, Canada, L8S 4K1**

**UDK 681.3.06[PROLOG:POPLOG]**

## ABSTRACT

There are many software tasks that could be solved more efficiently if they used both procedural and declarative programming techniques. One practical and useful solution is to combine the programming languages Prolog and POP-11 into one environment called POPLOG.

This paper discusses the ways of constructing a Prolog compiler in terms of POP-11 data structures and control techniques. Afterwards, the McMaster version of the Prolog compiler and the implementation of a few built-in Prolog procedures are introduced; moreover, the communication means between POP-11 and Prolog in Mc POPLOG (the McMaster version of POPLOG) are described.

## 1. INTRODUCTION

There exist many software engineering problems that utilize both procedural and declarative programming techniques. Several attempts to combine the programming language Prolog [3], [9] with procedural programming languages have been done such as:

- Prolog and LISP, e.g. LOGLISP [13], QLOG [10],
- Prolog and POP-11, i.e. POPLOG [11],
- Prolog and Modula, see [12].

We are convinced that the combination of POP-11 and Prolog is a very useful and adequate solution. Although both LISP and POP-11 [2] are AI programming languages, the programming language POP-11 provides more data structures and control techniques that can be used for implementation of Prolog terms and its inference machine. The first version of such a system that combines the languages POP-11 and Prolog has been proposed and implemented at Sussex University, U.K. This multi-language environment is called POPLOG [11].

Although the idea of joining two high-level AI programming languages into one system (POPLOG) is excellent, the actual Sussex implementation can be improved in several aspects. Therefore the author of this paper has decided to develop and implement a completely new version of POPLOG (Mc POPLOG) [5], [6] with these attributes:

- the entire McMaster version is written in C (rather than in POP-11),
- the system is very compact, since the Prolog subsystem utilizes only standard POP-11 data structures and control mechanisms without any change,
- it has straightforward communication means between both languages,
- the POP-11 subset has been substantially simplified by returning partly to the Edinburgh's POP-2 [7], taking new features of Sussex's POP-11 [2] that are important to AI projects (not to system tasks),
- the McMaster version is much shorter and faster (see Chapter 6 for a comparison), and is portable to any 32-bit machine.

This paper describes the way of developing a Prolog compiler in terms of POP-11 data structures and control techniques. Chapter 2 explains POP-11 structures necessary for such an implementation (see also [2], [7]). Chapters 3.1 and 3.2 introduce the fundamental model of a Prolog compiler written in POP-11 (details are in [11]). Chapters 3.3, 4, and 5 describe the McMaster model of a Prolog compiler in POP-11, the implementation of a few Prolog standard procedures, and the communication means between POP-11 and Prolog in the Mc POPLOG. Chapter 6 compares all three models (the fundamental, Sussex, and the McMaster ones).

## 2. CLOSURES AND NON-STANDARD CONTROL STRUCTURES OF POP-11

The programming language POP-11 [2] is a descendant of POP-2, a language originally developed at Edinburgh University for AI research in 1971 [7]. Now, there exist a few dialects of POP-2; the most famous are POP-11 (for PDP-11, VAX-11, and others) and WonderPOP (for DEC-10). POP-11 is mostly used for list processing, like LISP, but it uses Pascal-like (or Algol-like) well-structured syntax, has a large set of various data structures including records, vectors, strings, arrays, references, pairs, lists, dynamic lists, closures etc., and involves many non-standard control structures such as function-unwinding, processes, backtracking, pattern matching, database processing, macro facility, compiling-during-execution facility, and so on. Like BASIC, it is fully interactive, dialogue language. On the other hand, POP-11 has been mostly implemented as a compiler.

In this chapter, we introduce only closures and non-standard control mechanisms of POP-11 that are necessary for explaining how a Prolog compiler is written in POP-11. Other necessary syntax and built-in functions of POP-11 are in the Appendix. See also [2], [4], [7].

If a function, say `f(a,b,c,d)` is called often for some values `c = c0` , `d = d0` , we can create a new function `g(a,b)` so that

```
g(a,b) = f(a,b,c0,d0)
```

for all values `a` , `b` . This function is called a *closure* . A closure is defined in POP-11 so that the original function is followed by a sequence of actual arguments enclosed into decorated round parentheses `(%` and `%)` , i.e.

```
f (% c0,d0 %) -> g;
```

The arguments `c0` , `d0` are called *frozen values*, the function `f` is called a *frozen function*, and the entire process is called *partial application.*

One common use of the technique of closures is to save (or *freeze*) values of some variables at a certain time for a later call, if the variables are expected to change their values. This feature is especially utilized by a so-called continuation backtracking, discussed later.

The programming language POP-11 offers a large collection of non-standard control structures that can be used to construct efficient and elegant solutions of problems which would be inefficient to solve using convenient control structures. The Prolog compiler written in POP-11 needs the following control techniques: function-unwinding and backtracking.

1. *Function-unwinding.* If we want to quit a function and immediately call another function, the POP-11 function `chain` can be used. The function call `chain(G)` invoked within a function `F` unwinds the call of the function `F` and immediately calls the function `G` .

```
function alpha(x);
    ...
    beta(x);
    ...             /*here after executing function gamma */
end;                /* called by chain(gamma) within beta */

function beta(x);
    ...
    chain(gamma);   /*unwinds the call of  beta  and calls */
    ...             /* immediately the function gamma */
end;

function gamma;     /*called by function-unwinding mechanism*/
    ...
end;
```

If we call the function `alpha` , the function `beta` is invoked in a standard way but `chain(gamma)` causes the call of the function `beta` to be unwound, and the function `gamma` is performed. After executing the function `gamma` the control returns to the function `alpha` after the point where `beta` was invoked.

The function `chain` is used for the tail-optimization of Prolog procedures in POPLOG.

The function-unwinding mechanism of POP-11 can be applied not only for a single function call, but more than one call can be unwound. Here we need only these functions:

- the function `chainto(F,G)` unwinds all function calls up to (but excluding) the function `F` ; from here the function `G` is called in a standard way;
- `exitfrom(F)` unwinds all function calls up to and including the function `F` ; from here the execution continues in a standard way.

2. *Continuation backtracking.* The programming language POP-11 involves two types of backtracking that can be used for many AI applications. The first type, state-saving backtracking, has special keywords and constructs for saving nodes of a search tree. It is quite simple, but needs a large memory. See e.g. [1], [4], [7].

The other type, continuation backtracking, does not have special constructs, but only a special interpretation of function calls. In ordinary programming, a function call can be seen as a request to reach a goal. When a goal has been successfully performed, control returns to the invoker where the request was done, and continues. In the continuation backtracking, a function call is still seen as a request to achieve a goal. The difference is that the called function is told what should be done next if the goal is completed. This specification of what is to be done next is passed as an extra argument to the function, and is called *continuation.* Thus, the continuation backtracking has this interpretation:

- if the goal succeeds the function does not return to its invoker, but the continuation is applied;
- if the goal fails, the function returns to its invoker; thus normal return from a function is interpreted as a failure.

Example. Consider the following puzzle [1]. Each positive integer `n` (which is not divisible by 3) can be constructed by multiplying by 2, or adding 3, starting from 1, e.g.

$10 = (((\ 1 * 2\ ) * 2\ ) + 3\ ) + 3$

The function `alg(n)` solves this puzzle by continuation backtracking:

```
function alg(n); vars list;
    alg1(lambda; rev(list); exitfrom(alg) end,
            /*initial continuation*/
        1, nil)
end;

/*here  n1  is the current number, initially 1 ,
 list  involves a sequence of actions, initially  nil */

 function alg1(contin,n1,list);
     if n1 == n then
        contin()            /*success -so continue */
     endif;
     if n1 < n then
                            /*firstly try goal *2 */
        alg1(contin, n1*2, 2::"*"::list);
                            /*fail:  try +3 */
        alg1(contin, n1+3, 3::"+"::list)
     endif                  /*if n1>n then return:  failure */
 end;

 alg(10) =>

 ** [ * 2 * 2 + 3 + 3]
```

If a goal comprises two subgoals then in ordinary programming, we call both subgoals in a sequence. E.g. let a goal (function) `g(x)` have subgoals `a(x)` and `b(x)` . Then in ordinary programming:

```
function g(x);
    a(x);                   /*call the 1.subgoal*/
    b(x);                   /*then call the 2.subgoal*/
end;
```

In continuation backtracking, the subgoal `b` will be passed as a continuation argument to the function `a` :

```
function g(contin,x);
    a( b(% contin, x %), x)
end;
```

So, if we invoke g we have to supply explicit continuation as to what is to be done when the goal g is successfully accomplished. The function g invokes the subgoal (function) a with the continuation

```
b(% contin, x %)
```

Consequently, if the function a succeeds, the above continuation will be invoked, i.e., the function b will be invoked with the original continuation and the frozen value of x .

Thus, a sequence of goals in the continuation backtracking is realized by the closure mechanism. No other constructs are necessary. The continuation backtracking is obviously faster (2 to 4 times, task-dependent) than the state-saving backtracking and does not need so much memory. Therefore, this type of backtracking is the basis for the Prolog implementation in POP-11.

## 3. A PROLOG COMPILER WRITTEN IN POP-11

### 3.1. Prolog variables

Any implementation of Prolog must solve many problems, among them the most important ones seem to be:

- unification,
- backtracking,
- instantiation of Prolog variables,
- the 'cut' operator.

Evidently, the only type of backtracking suitable for Prolog implementation in terms of POP-11 is the continuation backtracking. This technique is utilized without any change. The 'cut' operator can be implemented by utilizing the POP-11 function-unwinding techniques. In this chapter, we survey how the remaining problems can be solved. Other interesting ideas of Prolog implementations can be found in [8].

Unfortunately, a Prolog variable cannot be directly represented by a POP-11 variable, because we could not express the matching of two uninstantiated Prolog variables in such a representation. Moreover, there would also arise problems with the backtracking. The only adequate representation of a Prolog variable seems to be a POP-11 reference; it is a standard record of one component (see Appendix).

(1) An uninstantiated variable X is represented by a reference containing the word undef . This is done by the POP-11 function consref :

```
consref(undef) ->X;
```

i.e., a reference with contents undef is assigned to X .

(2) If an uninstantiated variable X matches a term T , then the value of T is attached to the variable X . In POP-11, this is done by the accessor cont that can select or update contents of any reference:

```
T ->cont(X);
```

(3) If two uninstantiated variables X and Y share then one reference becomes the contents of the other, e.g.

```
Y ->cont(X);
```

i.e. the Prolog variable (POP-11 reference) X contains a pointer to the Prolog variable Y and both of them are uninstantiated. This *chaining* of shared uninstantiated variables can be done to any depth. Therefore, if we process Prolog variables, we should *dereference* them, i.e., look at the contents of the innermost reference. For that purpose, we use the auxiliary function deref(X).

### 3.2. Fundamental model with explicit re-uninstantiation

The unification algorithm for pattern matching is one of the most important mechanisms of Prolog. To demonstrate fundamental ideas of compiling Prolog procedures, the POP-11 function unify on Fig. 1 can be introduced as a unification algorithm (see [11] for details). It assumes Prolog constants are POP-11 numbers or words, Prolog variables are POP-11 references (with recognizer isref, and accessor cont), and Prolog structures are POP-11 pairs (with recognizer ispair, and accessors front, back). The last statement with the unify call tries to unify front(X) and front(Y) , and if it succeeds, back(X) and back(Y) are unified. If even they match, the given continuation contin is invoked.

The above model of unification undoes any changes it has made when it returns to its invoker, i.e. when a fail takes place, see the statements for re-uninstantiation:

```
undef ->cont(X) , undef ->cont(Y)
```

However, the representation and matching of a general structure f(a1,a2,...,an) with functor f , arity n , and components a1,a2,...,an is unclear. One way to represent internally such a structure is as a list

```
[ f a1 a2 ...  an ]
```

but it would be very cumbersome and time-consuming. Note the actual models do not use such an internal representation of Prolog structures.

If we accept the above function unify as a fundamental model of unification, then Prolog clauses can be easily compiled to POP-11 functions. In the following, we discuss how Prolog facts and rules with conjunctions and disjunctions of goals are compiled. See [11] for details.

A Prolog fact is compiled by means of the function unify. If there are more facts of the same predicate in the Prolog database the *return-fail* mechanism of the continuation backtracking is simply utilized. E.g. let the database contain

```
girl(silva).
girl(paulina).
```

then the predicate girl is compiled as the following POP-11 function:

```
function girl(contin,X);
                /*1.girl:  "silva" is POP-11 word*/
    unify(contin, X, "silva");
                /*fail:  try another girl*/
    unify(contin, X, "paulina")
end;
```

A conjunction of two or more goals in a body of a rule is compiled by means of closures as continuations. Consider a conjunction of two goals:

```
likes(X) :- girl(X), drinks(X).
```

then the second goal will be placed as a continuation for the first goal, i.e. the above procedure is compiled as

```
function likes(contin,X);
    girl( drinks(%contin, X%), X)
end;
```

```
function unify(contin,X,Y);
    deref(X) ->X;  deref(Y) ->Y;
    if X == Y then                      /*unify constants*/
        contin()
    elseif isref(X) then                /* X  is uninstantiated var*/
        Y ->cont(X);                    /* X  instantiated to  Y */
        contin();                       /*success: continue*/
        undef ->cont(X)                 /*fail:  X  re-uninstantiated var*/
    elseif isref(Y) then                /* Y  is uninstantiated var*/
        X ->cont(Y);
        contin();
        undef ->cont(Y)                 /*fail:  Y  re-uninstantiated var*/
    elseif ispair(X) and ispair(Y). then
                                        /* X , Y  structures of the same type*/
        unify( unify(% contin, back(X), back(Y) %),
            front(X), front(Y) )
    endif                               /*otherwise fail (no 'else' here) */
end;
```

Fig. 1. Function unify .

The function `girl` is called to unify `X` with a girl in the database. Assume Prolog database contains two girls `silva` and `paulina` . The function `girl` succeeds with `X = silva` . Afterwards, the function `drinks` is called as a continuation within the function `girl` . If the second goal succeeds, i.e. `silva` drinks, then the given continuation `contin` is invoked. If the girl `silva` does not drink (more precisely, the system does not have an information about her habits), then the function `drinks` returns to its invoker, and another girl is processed. If no girl drinks then the call of `girl` returns to `likes` , and `likes` fails immediately.

If more than two goals are in a conjunction in a clause body then closures as continuations must be nested. E.g.

```
likes(X) :- girl(X), drinks(X), slim(X).
```

is compiled as

```
function likes(contin,X);
    girl( drinks(% slim(% contin, X%), X%), X)
end;
```

A disjunction of goals, or a sequence of clauses for one procedure is compiled by utilizing the *return-fail* mechanism of the continuation backtracking. E.g. let the procedure `drinks(X)` be defined by these two clauses:

```
drinks(X) :- wine(X).
drinks(X) :- beer(X).
```

then the corresponding POP-11 function is

```
function drinks(contin,X);
    wine(contin, X);     /* X drinks wine*/
    beer(contin, X);     /*fail:  try beer */
end;
```

Example. Prolog procedure `member(X,L)` defined as

```
member(X,[X| ]).
member(X,[_|T]) :- member(X,T).
```

can be compiled as the following POP-11 function, using only the function `unify` :

```
function member(contin,X,L); vars T;
            /*1.clause:  2.arg unified with [X|_] */
    unify(contin, L, X::consref(undef));
            /* ::  is constructor for pairs*/
            /*1.clause fails:  try the 2.clause*/
    consref(undef) ->T;
            /*2.arg unified with [_|T] */
    unify(member(%contin,X,T%), L, consref(undef)::T)
end;
```

We can see that

- the way of compiling Prolog procedures is very simple, straightforward,
- the result of compilation, i.e. a POP-11 function is slow in execution,
- the principal disadvantage of this model is that structures are constructed many times only for the purposes of the head matching, e.g.

```
X ::  consref(undef)   for matching  [X|_]
consref(undef) :: T                  [_|T]
```

The Sussex version of POPLOG uses a very detailed compilation of head matching (see [11], pp. 161-162) so that no structures are constructed for the purposes of head matching. It represents precisely the idea of the continuation backtracking and compiled Prolog procedures as POP-11 functions are fast in execution. A comparison of all models is introduced in Chapter 6.

## 3.3. Auxiliary stack of instantiated Prolog variables

Mc POPLOG does not use the detailed way of compilation but rather utilizes a set of auxiliary functions that recognize a 'type' of a processed argument (i.e. whether the argument is a constant, uninstantiated variable, or a structure). Furthermore, the mechanism of instantiating Prolog variables is using an auxiliary stack, rather than assignment statements for explicit re-uninstantiation.

McMaster model [6] uses an auxiliary stack for storing instantiated Prolog variables (so-called *Prolog variable stack*). The current offset to the Prolog variable stack is designated `varoff`. If a Prolog variable is instantiated then it is pushed onto the Prolog variable stack. Secondly, when a branching point in a search tree has been reached, the system saves the current offset of the Prolog variable stack in the variable `varoff1` .

If a fail occurs, then the system returns to the branching point and re-uninstantiates all Prolog variables that have

```
function punify(X,Y);  vars i;
    deref(X) ->X;  deref(Y) ->Y;
    if X == Y then                      /*identical terms: result is true */
        true
    elseif isref(X) then                /* X  is uninstantiated var*/
        Y -> cont(X);                   /*assign term  Y  to var  X */
        X ->varstack(varoff);           /*push  X  onto the Prolog var stack*/
        varoff + 1 ->varoff;
        true                            /*result is true: matching succeeded*/
    elseif isref(Y) then                /* Y  is uninstantiated var*/
        X -> cont(Y);                   /*assign term  X  to var  Y */
        Y ->varstack(varoff);           /*push  Y  onto the Prolog var stack*/
        varoff + 1 ->varoff;
        true                            /*result is true: matching succeeded*/
    elseif ispair(X) and ispair(Y) then
                                        /*both terms are pairs*/
        punify(front(X),front(Y)) and punify(back(X),back(Y))
    elseif isvector(X) and isvector(Y)
        and datalength(X) == datalength(Y) then
                                        /*both term are general structures of*/
                                        /* the same arity*/
        for i from 1 to datalength(X) do
            ifnot punify(X(i),Y(i)) then    /*unify i-th components*/
                false.return
            endif
        enddo;
        true                            /*functors & all components match*/
    else
        false                           /*otherwise result is  false */
    endif
end;


function passign(X,T);                  /*assign term  T  to Prolog var X */
    T ->cont(X);
    X ->varstack(varoff);               /*push  X  onto the Prolog var stack*/
    varoff + 1 ->varoff
end;
```

Fig. 2. Functions punify and passign .

```
function member(contin,X,L);
vars T, varoff1;                        /*local variables*/
   varoff ->varoff1;                    /*save current offset to the */
                                        /* Prolog variable stack*/
   deref(X) ->X;  deref(L) ->L;
                                        /*1.clause: head matching*/
   if isref(L) then                     /* L  is uninstantiated var*/
      passign(L, X::consref(undef))     /* [X | _]  is assigned to  L */
   elseif ispair(L) then                /* L is a pair*/
      ifnot punify(X,front(L)) then     /* L = [X | _]  ? */
         goto clause2
      endif
   else  goto clause2
   endif;
   contin();                            /*no body of the 1.clause- so continue*/
clause2:                                /*fail: 2.clause*/
   retrieve();                          /*uninstantiate Prolog vars*/
                                        /*head matching*/
   if isref(L) then                     /* L  is uninstantiated var*/
      consref(undef) -> T;              /*new Prolog var to  T */
      passign(L, consref(undef)::T)     /* [_ | T]  is assigned to  L */
   elseif ispair(L) then                /* L  is a pair*/
      back(L) -> T                      /* L = [_ | T] */
   else  goto clause3
   endif;
                                        /*body optimized: quit this call and */
   chain(contin,X,T,member)             /*call immediately member(contin,X,T) */
clause3:
end;
```

Fig. 3. Procedure member compiled by McMaster Prolog compiler.

been instantiated after the branching point has been encountered. The re-uninstantiation is done by the auxiliary function `retrieve()` that uninstantiates all Prolog variables in the Prolog variable stack between the current offset `varoff` and the initial offset `varoff1` :

```
function retrieve;
    while varoff > varoff1 do
        varoff - 1 -> varoff;
        undef -> cont(varstack(varoff))
    enddo
end;
```

Note the Prolog variable stack is implemented as a POP-11 vector, and `varstack` is the bottom of this stack.

The unification algorithm of Mc POPLOG is written as an ordinary function, without the continuation mechanism. The function `punify(X,Y)` (see Fig. 2) returns `true` iff the terms `X` and `Y` match, and instantiates Prolog variables if necessary. Note that a Prolog structure `f(a1,a2,a3)` is represented as the POP-11 vector `{ f a1 a2 a3}`.

Besides the above unification function, few other functions can be invoked within compiled Prolog predicates. Among them, the function `passign(X,T)` assigns the term `T` to the Prolog variable `X` : see Fig. 2.

Example. The procedure `member(X,L)` is compiled in McMaster Prolog as follows: see Fig. 3.

The McMaster model of a Prolog compiler
- follows the idea of the continuation backtracking,
- constructs no structures in the head of any clause,
- Prolog structures are represented more efficiently as POP-11 vectors,
- compiled Prolog procedures as POP-11 functions are fast in execution, and not so large,
- the system does not need so large system stack (for storing function calls) since many auxiliary functions of this model are written as convenient functions without the continuation backtracking,
- Warren's tail optimization is always done.

Note that in the actual implementation, the functions `deref`, `isref`, and `ispair` used above are joined into one auxiliary function so that the execution is even faster.

A comparison of the all three models is done in Chapter 6.

## 4. COMPACT IMPLEMENTATION OF BUILT-IN PROLOG PROCEDURES

The Prolog subsystem of the Mc POPLOG uses the above model with the Prolog variable stack, the functions `punify(X,Y)`, `passign(X,Y)`, `deref(T)` etc., and Prolog variables are represented as POP-11 references. We will now describe how built-in Prolog procedures are implemented in this model.

First of all, the unification operator `=` invokes directly the function `punify`; its POP-11 representation is on Fig. 4. The operator `is` uses the function `punify`, and the function `popval(P)` that evaluates the Prolog term `P` as a POP-11 expression (see also Chapter 5.2); its POP-11 representation is on Fig. 4.

Fig. 4 involves some other built-in procedures that can be written in terms of the operator `is`, or they utilize the functions `punify` or `popval` (see also Chapter 5.2 for details).

The operators for constructing goals and backtracking are also quite simple. Among them, Fig. 5 introduces `';'` and `repeat`. The function `papply(contin,X)` used here considers the term `X` as a goal, and applies it with the given continuation.

The 'cut' operator `'!'` is considered as the goal

```
cut_1(caller(1))
```

where `caller(1)` is the invoker of the current function, i.e. the parent goal of the goal that contains `'!'` in its body; e.g. if

```
g :- a, b, c.
b :- d, e, !, f.
```

then the above 'cut' operator is considered as `cut_1(g)`. The function `cut_1` calls `chainto` :

```
function cut_1(contin,F);
      chainto(F,contin)
end;
```

## 5. COMMUNICATION MEANS BETWEEN POP-11 AND PROLOG

### 5.1. Top level

The Mc POPLOG allows the user to call POP-11 functions from Prolog and vice versa. This inter-language communication can be done to any depth.

When the user is at the top level of the system POPLOG then the prompt `:` is displayed and the system expects any POP-11 statement or any Prolog question. If a text begins with `?-` then this `?-` is recognized as the operator of a Prolog question, otherwise it is considered as a POP-11 statement. Therefore, typing in `?-` followed by a Prolog question will activate the Prolog subsystem. E.g.

```
:  member(2,[1 2 3]) =>        /*POP-11 statement:  is 2 a member
                                of the list [1 2 3] ?  */
** true                        /*answer of POP-11 subsystem*/
:
:  ?- read(X),write(X),nl.     /*Prolog question*/
| maria.                       /*term read by read(X) */
maria                          /*printed by Prolog's write(X)*/
X = maria                      /*instantiated variable displayed*/
yes                            /*answer of Prolog subsystem*/

:                              /*prompt of POPLOG is waiting for
                                next statement or question*/
```

Prolog clauses can be asserted into the Prolog database at the top level using the standard procedure `consult(user)` or `[user]` . POP-11 statements and Prolog questions and clauses can be also used together in a file.

### 5.2. Calling POP-11 functions from Prolog

The user of Mc POPLOG can call any POP-11 function from within Prolog. For that purpose, semantics of the standard Prolog operator `is` has been extended within the system POPLOG, and two additional unary procedures `popval` and `popvalfail` have been supplied.

`popval(P)` evaluates the Prolog term `P` as a POP-11 expression. The expression `P` must not return any result. This goal succeeds only once.

`popvalfail(P)` evaluates the Prolog term `P` as a POP-11 expression; one result must be returned, and if it is POP-11's `false` it is interpreted as a fail of the given goal. This goal

```
function eq_2(contin,X,Y);           /* X = Y */
    if punify(X,Y) then
        contin()
    endif
end;


function is_2(contin,X,Y);           /* X is Y :- X = popval(Y) . */
    if punify(X,popval(Y)) then
        contin()
    endif
end;


function lt_2(contin,X,Y);           /* X < Y */
    if popval(X) < popval(Y) then
        contin()
    endif
end;


function var_1(contin,X);            /* var(X) :-  */
                                     /*  popvalfail(isref(closure(X))) */
    if isref(deref(X)) then          /*if  X  is uninstantiated var*/
        contin()
    endif
end;


function functor_3(contin,T,F,Ar);   /* functor(T,F,Ar) */
vars i;
    deref(T) ->T;  deref(F) ->F;  deref(Ar) ->Ar;
    if isnumber(T) then              /* T  is a number*/
        punify(T,F) and punify(Ar,0)
    elseif isref(T) then             /* T  is uninstantiated var*/
        punify(T, {% F,              /* T  is unified with  F(_,_, ... ,_) */
                    for i from 1 to Ar do
                       consref(undef)
                    enddo %})        /*Prolog structures are POP-11 vectors*/
    else                             /* T is nonvar */
        punify(F, T(1)) and          /* F  is unified with functor of  T */
            punify(Ar, datalength(T)-1)  /* Ar is unified with arity of T */
    endif;
    if /*result of punify*/ then     /*if matching has succeeded continue*/
        contin()
    endif
end;
```

Fig. 4. Implementation of some built-in Prolog procedures

```
function or_2(contin,X,Y);           /* X ; Y */
vars varoffl;
    varoff -> varoffl;               /*save the current offset to Prolog */
                                     /* var stack*/
    papply(contin,X);                /*call the 1.goal*/
    retrieve();                      /*fail: uninstantiate Prolog vars*/
                                     /*try the 2.goal with the tail */
    chain(contin,Y,papply)           /* optimization: quit this call and */
end;                                 /*call immediately  papply(contin,Y) */


function repeat_0(contin);           /* repeat */
vars varoffl;
    varoff -> varoffl;               /*save the current offset to Prolog */
repl:                                /* var stack*/
    contin();
    retrieve();                      /*fail: uninstantiate Prolog vars*/
    goto repl;                       /*loop*/
end;
```

Fig. 5. POP-11 representation of  ';'  and  repeat .

succeeds only once.

`X is P` evaluates the Prolog term `P` as a POP-11 expression and unifies the result with `X` ; `P` must return just one result.

The Prolog term `P` in the above procedures ( `popval`, `popvalfail`, `is`) is comprehensibly written in Prolog syntax because the Prolog compiler reads this text. However, it is evaluated as a POP-11 expression. Here are the rules for *POP-11 evaluation of Prolog terms* (the rules (iii), (iv) will be explained later):

(i) if `P` is a constant (or an instantiated Prolog variable) then the constant (value of the Prolog variable) is pushed onto the POP-11 user stack;

(ii) if `P` is an uninstantiated Prolog variable then its POP-11 representation (a reference) is pushed onto the POP-11 user stack (see the representation of Prolog terms as POP-11 items);

(iii) `closure(P)` where `P` is a Prolog term is evaluated by pushing `P` onto the POP-11 user stack;

(iv) `V -> A` where `V` is an instantiated Prolog variable, `A` is Prolog atom, is evaluated as follows: the value attached to the Prolog variable `V` is assigned to POP-11 variable `A` ;

(v) the Prolog structure `f(P1,...,Pn)` where `f` is n-ary functor and `P1` to `Pn` are Prolog terms, is evaluated as follows: firstly `P1` to `Pn` are evaluated as POP-11 expressions and then the POP-11 function `f` is called.

Thus, the *POP-11 evaluation of a Prolog term* `P` matches the standard evaluation of POP-11 expressions in the programming language POP-11 [2].

Note the *POP-11 evaluation of a Prolog term* `P` does not affect the POP-11 user stack because

- `popval(P)` must not return any result,
- `P` of `popvalfail(P)` returns just one result that is popped from the stack, and its value decides about success/failure of the goal,
- in case of `X is P` the value of `P` is popped from the stack and unified with `X`.

If *POP-11 evaluation of a Prolog term* does not return the required number of results, a run-time error occurs.

Example. Define the Prolog procedure `concat(L1,L2,L3)` (`L3` is a concatenation of the lists `L1, L2`) for the instantiated `L1` and `L2`. We will use the standard POP-11 function `join` for list concatenation:

```
concat(L1,L2,L3) :- L3 is join(L1,L2).
```

Now

```
?- concat([1,2], [a,b], R).
R = [1, 2, a, b]
yes
```

However, the question

```
?- concat([1,2], [aa(1),bb], R).
```

would cause an error since the entire argument `join([1,2],[aa(1),bb])` would be considered as a POP-11 expression so that a non-existing POP-11 function `aa` would be called with the argument `1` , see Fig. 6. To prevent a subexpression of an argument of the procedures `popval`, `popvalfail`, `is` to be considered as a POP-11 expression one has to *close* it using the functor `closure` as follows:

```
concat(L1,L2,L3) :- L3 is join(closure(L1), closure(L2)).
```

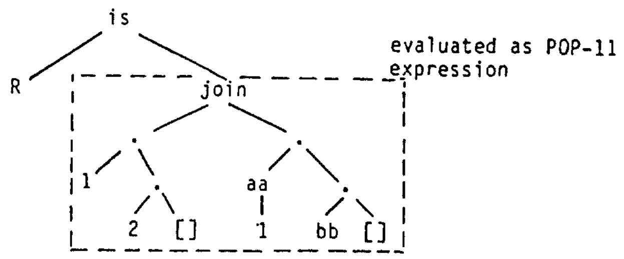


Fig. 6. The body of the procedure concat after matching (lists are expressed as dot-pairs)

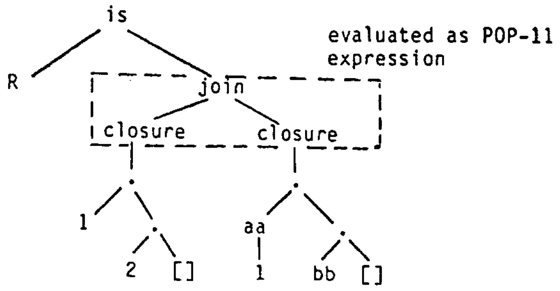


Fig. 7. The arguments of the function join are 'closed' now

Now `L1, L2` are considered as Prolog terms rather than POP-11 expressions, see rule (iii). Thus

```
?- concat([1,2], [aa(1),bb], R).
R = [1, 2, aa(1), bb]
yes
```

will return expected results, see Fig. 7.

Example. Consider Prolog procedure `member(X,L)` : `X` is an element of the list `L` . If the procedure is used only for instantiated `X` and `L`, then - in some cases - it could be defined in terms of the standard POP-11 function `member` :

```
member(X,L) :- popvalfail(member(closure(X),closure(L))).
```

To process a value of a POP-11 variable within a Prolog program the standard POP-11 function `valof` (value of a variable) has to be used:

```
:  vars list;              /*declare a POP-11 variable*/
:  [10 9 a ab aaa 0] ->list; /*assign the given list to it*/
:                          /*sort list by Prolog's sort */
```

If we wrote

```
?- List is list, sort(List,S).
```

the Prolog variable `List` would be instantiated to the atom `list`. To introduce a value of the POP-11 variable `list`, we must do:

```
: ?- List is valof(list), sort(List,S), write(S), nl.
[0, 9, 10, a, aaa, ab]        /*printed by write(S), nl */
List = [10, 9, a, ab, aaa, 0] /*instantiated variables */
S = [0, 9, 10, a, aaa, ab]    /* displayed*/
yes
```

The evaluation of `valof` is embodied in the rule (v).

A POP-11 expression in `popval, popvalfail, is` can also contain an assignment to a POP-11 variable, using the POP-11 operator `->` , see rule (iv).

Example.

```
:  vars list, sorted;
:  [10 9 a ab aaa 0] ->list; /*sort list by Prolog's sort */
:
:  ?- List is valof(list), sort(List,S), popval(S ->sorted).
                              /*result of sort assigned to */
                              /* POP-11 variable sorted */
List = [10, 9, a, ab, aaa, 0]
S = [0, 9, 10, a, aaa, ab]
yes

:  sorted =>                  /*POP-11 variable sorted is */
                              /* the sorted list*/
** [ 0 9 10 a aaa ab]
```

Thus, thanks to `valof` and `->` we can use POP-11 variables as *inputs* to Prolog programs, as well as *outputs* for storing results.

### 5.3. Prolog called by POP-11 functions

Similarly, any POP-11 function can call a Prolog program by introducing a Prolog question in its body. When a Prolog question is executed within a POP-11 function

- the Prolog subsystem does not wait for user's replies;
- but returns `true` ( `false` ) as a result, if the Prolog question has succeeded (has failed);
- all Prolog variables in the Prolog question will retain their values when returning from the Prolog question to the POP-11 function so that they can be processed as any other POP-11 variable.

Example. Define POP-11 function `sort(list)` using the standard Prolog procedure `sort(L1,L2)` :

```
function sort(list) => R;
                          /* R is explicit result of function*/
   ?- List is valof(list), sort(List,R).
                          /* R retains its value from Prolog */
                          /* question*/
end;
```

Now, the POP-11 expression `sort([10 9 a ab aaa 0])` yields

```
[ 0 9 10 a aaa ab]
```

Example. Define POP-11 function `member2(x,l)` using the Prolog procedure `member(X,L)` defined as

```
member(X,[X|_]).
member(X,[_|T]) :- member(X,T).
```

We utilize directly a result returned by the Prolog question:

```
function member2(x,l);
   ?- X is valof(x), L is valof(l), member(X,L).
end;
```

If the goal `member(X,L)` succeeds then `true` is returned, otherwise `false` . Thus `true / false` becomes the result of the POP-11 function `member2` .

## 6. CONCLUSION

It is important to emphasize that in order to economize time and memory requirements of the system, the entire Prolog subsystem of Mc POPLOG has been written in C rather than in POP-11. We have introduced the ideas in terms of POP-11 functions for better and easier readability.

Secondly, a Prolog compiler could translate Prolog clauses to POP-11 functions, as we have shown above, and afterwards call the POP-11 compiler which would translate a text of a POP-11 function to a machine code. In the actual implementation, the Prolog compiler translates any Prolog procedure directly to the machine code. All the above examples of compiling Prolog procedures are written in POP-11 to simplify the explanation of principal ideas.

It was not a straightforward procedure to compare the above three models (the fundamental, Sussex, and McMaster one) of Prolog compilers using the POP-11 structures, since

- the fundamental model (Chapter 3.2 and [11]) is not a part of any POPLOG version,
- we had at disposal the executable (binary) file of the Sussex POPLOG and [11] only (not a source file, nor any detailed information).

Therefore:

- we have emulated the fundamental model in POP-11 (McMaster version),
- we have compiled the procedure `member` on the emulator, Sussex version, and McMaster version,
- we have run the question

```
?- member(100, [1,2,3, ... ,100]).
```

The entire comparison is in Fig. 8. As the timings depend on the percentage of calls of standard and user-defined procedured, we have added the execution time of the following question, as well:

```
?- functor(T,aa,100).
```

The results presented in Fig. 8 match quite well a long term statistical experiments we have done for the comparison of both actual implementations of POPLOG; see Fig. 9.

We characterize the McMaster version of POPLOG as a compact one since the Prolog subset utilizes only standard POP-11 data structures, as well as POP-11 control mechanisms, without any change. Thanks to that approach, the Mc POPLOG is unbelievably shorter, and also faster.

The communication means between POP-11 and Prolog in the Mc POPLOG are quite straightforward and simple. Neither programming language has been changed so that a user unfamiliar with POP-11 can use the Prolog subset only, and vice versa. Only three new predicates (`popval, popvalfail, closure`) have been added to the Prolog subset, and the semantics of Prolog's `is` has been slightly extended.

The Mc POPLOG runs currently on VAX systems and on systems based on the Motorola 68000 processor. Detailed information on the system can be obtained from the author.

| | emulator of the fundamental model | Sussex POPLOG | Mc POPLOG |
|---|---|---|---|
| compilation time | 8 | 38 | 4 |
| # of machine instructions for member | 21 | 60* | 39 |
| execution time of the question ?- member ... | 127 | 35 | 15 |
| execution time of the question ?- functor ... | - | 42 | 12 |

Fig. 8. Comparison of the Prolog compiler models written in POP-11
Compilation and execution time is in cs (centiseconds), i.e. hundredths of seconds
The emulator of the fundamental model has been tested in McMaster version of POPLOG
* Estimate done according to the example in [11], pp. 161-2

| | McMaster | Sussex | Comparison: |
|---|---|---|---|
| (1) the system needs memory | 75 KB | 670 KB | 9 times shorter |
| (2) POP-11: | | | |
| - compilation time | | | 3 times faster |
| - execution time | | | 2 times faster |
| (3) Prolog: | | | |
| - compilation time | | | 3.7 times faster |
| - execution time | | | 2.7 times faster |

Fig. 9. Memory requirements and timings of the actual versions of POPLOG

## REFERENCES

[1] Anderson, B.: Programming languages for AI: the role of nondeterminism. School of AI, Rep. 25, Univ. of Edinburgh, 1972

[2] Barrett,R., Ramsay,A., Sloman,A.: POP-11 - a practical language for artificial intelligence. Ellis Horwood, New York, 1985

[3] Bratko,I.: Prolog - programming language for artificial intelligence. Addison-Wesley, 1986

[4] Bruha, I.: User's guide to POP-11 of McMaster POPLOG. Dept. Computer Science and Systems, McMaster Univ., Techn. report 87-04, 1987

[5] Bruha, I.: Reference manual of McMaster POPLOG. Dept. Computer Science and Systems, McMaster Univ., Techn. report 87-03, 1987

[6] Bruha, I.: Compact implementation of Prolog as a part of the environment of McMaster POPLOG. Dept. Computer Science and Systems, McMaster Univ., Techn. report 87-05, 1987

[7] Burstall,R., Collins,D., Popplestone,R.: Programming in POP-2. Edinburgh University Press, 1971

[8] Campbell,J.A. (editor): Implementations of Prolog. Ellis Horwood, New York, 1984

[9] Clocksin,W.F., Mellish,C.S.: Programming in Prolog. Springer-Verlag Berlin, 1984

[10] Komorowski, H.J.: QLOG - The programming environment for Prolog in LISP. In: Clark, K.L. and Tarnlund, S.: Logic Programming, Academic Press, 1982

[11] Mellish,C., Hardy,S.: Integrating Prolog in the POPLOG environment. In: [8].

[12] Muller, C.: Modula - Prolog: A software development tool. IEEE Software, pp. 39-45, Nov. 1986

[13] Robinson, J.A. and Sibert, E.E.: LOGLISP: An alternative to Prolog. In: Machine Intelligence 10, Ellis Horwood, 1982

## Appendix

To understand the POP-11 programs introduced here we should know only:

`vars x, y;`
declaration of POP-11 variables `x, y`

`e ->x;`
value of the expression `e` is assigned to the variable `x`

`e =>`
value of the expression `e` is printed and preceded by `**`

`function f(x,y); ... end;`
declaration of the function `f` with formals `x, y`

`lambda (x,y); ... end;`
declaration of a no-name function (lambda-expression) with formals `x, y`

`ifnot C then ...`
the same as `if not(C) then ...`

`" ... "`
delimiters of a POP-11 word; thus `alpha` is a variable whereas `"alpha"` is a word (atom)

`isnumber(x)`
returns `true` iff the item `x` is a number

`consref(x)`
constructs a reference with contents `x`

`cont(r)`
function that selects or updates the contents of the reference `r`

`isref(x)`
returns `true` iff the item `x` is a reference

`front(p) / back(p)`
function that selects or updates the first/ second component of the pair `p`

`ispair(x)`
returns `true` iff the item `x` is a pair

`member(x,xl)`
returns `true` iff `x` is an element of the list `xl`

`rev(xl)`
returns a list of the elements of the list `xl` in the reverse order

`x1 :: x2`
operation that creates a pair from `x1` and `x2`

`join(xl,yl)`
returns the concatenation of the lists `xl` and `yl`

`{ ... }`
brackets for a vector construct

`s(i)`
the `i`-th component of the structure `s`

`datalength(s)`
returns the number of components of the structure `s`

# ON DELETION OF REFLEX ELEMENTS FROM CIRCULAR DOUBLY-LINKED LISTS

INFORMATICA 4/1988

UDK 519.683

**Mirjana Stojanović, Ivan Stojmenović**
**Institute of Mathematics,**
**University of Novi Sad**

ABSTRACT

An element x of a doubly-linked circular list is called reflex if f(pred(x),x,succ(x)) $\leq 0$. We propose an algorithm to delete reflex elements from the list and apply algorithm to give the correct code of Graham 's algorithm for determining the convex hull of a finite set of points in the plane. We show that the code given by Preparata, Lee and Shamos was incorrect and give some corrections to the results of Koplowitz and Jouppi.

Key words and phrases: convex hull, algorithm, circular list CR I.3.5 Computational geometry and object modeling
AMS Mathematical subject classification(1980): Primary 86U05, Secondary 68Q10.

## 1. Deletion of reflex elements from circular doubly-linked lists

Suppose we are given a circular doubly-linked list S and a function f(pred(x),x,succ(x)) the value of which for given element x of S depends on its predecessor pred(x) and successor succ(x) as well. The element x is called reflex if f(pred(x),x,succ(x)) $\leq 0$. The following algorithm will delete all reflex elements from the input list S, starting at a given initial element START. The result is stored in the list CH(S), initially equal to S. Although in general the result depends on the choice of START, in applications such as one given in the next section the output is the same for any choice of initial element.

```
BEGIN
      CH(S) = S;
      h:=false;
      g:=false;
      v:=START;
      r:=succ(r);
      REPEAT
           IF f(v,succ(v),succ(v)))> 0
                  THEN BEGIN {successful test}
                  IF r=succ(v) and hTHENg:=true;
                   v:=succ(v);
                   IFsucc(v)=rTHENh:=true
               END
         ELSE BEGIN
                   IFr=succ(v)THENr:=pred(v)
                   delete succ(v) from CH(S);
                   v:=pred(v)
               END
      UNTIL g
END.
```

In the algorithm, h is true iff all elements of CH(S) are examined as a middle point, while g is true iff h is true and there is an element exmined twice as a middle point. Finally r is the farthest predecessor (in CH(S)) of element to be tested next which had already a successful test (if any, or element to be tested next otherwise).

h will became true if, after one successful test, next element to be tested is r (i.e. the element examined first among remaining elements). A test should be performed again on r (since the predecessor of r may change) and,if successful, will terminate the algorithm (by assigning g to true). Otherwise r is moved to its predecessor and is equal to the currently tested element until a successful test id found. The technique is simple but not trivial ([2,3, 4] failed to apply it correctly in case of Graham's scan).

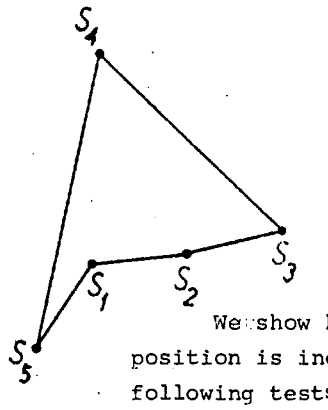


Fig. 1

## 2. The correct code of Graham's convex hull algorithm

Given a set S of n points in the plane,the convex hull CH(S) of the set is the smallest convex polygon containing the set. Graham [1] proposed an algorithm for determining the convex hull of a finite set of points in the plane that runs in O(n log n) time. Variations of his method giving increasing efficiency have been described by Koplowitz and Jouppi, and by Preparata, Lee and Shamos. In this section we give some corrections to their results. A correct code is given by means of the algorithm for deletion of reflex elements from doubly-linked circular list.

After constructing a set of points S in polar coordinates ordered about an interior point O in terms of increasing angle, and initially placed in CH(S), Graham's scan algorithm [1] repeatedly examines triples of consecutive points $S_1,S_2,S_3$ in counterclockwise order to determine whether or not they define a reflex angle (an angle $\geq \pi$). If an internal angle $S_1S_2S_3$ is reflex then the test is unsuccessful and $S_2$ cannot be extreme point because it is internal to triangle $OS_1S_3$. The point $S_2$ will be eliminated from CH(S) in this case and the next test is $S_0S_1S_3$. If the test is successful then $S_2$ is a convex hull candidate and a single scan around the ordered points will advance by checking $S_2S_3S_4$.

The correct code for the Grahman's scan can be obtained by specifying the function f(pred(v),v,succ(v)) in the algorithm in Section 1 to be angle equal to the difference between $\pi$ and angle formed by pred(v),v and succ(v). The angle at v is reflex iff v is reflex under our definition of f.

Koplowitz and Jouppi [2] claimed the following.

PROPOSITION. If all the points in CH(S) have been tested as a middle point and the last test performed has been a successful forward test, then the algorithm is complete.

We show by a counterexample that this proposition is incorrect. In Fig. 1 we have the following tests :

$S_1S_2S_3$, $S_2S_3S_4$, $S_3S_4S_5$,
$S_4S_5S_1$, $S_5S_1S_2$, $S_4S_5S_2$.

Only the fifth test is unsuccessful. Thus only the point $S_1$ will be eliminated from CH(S). All the remaining points $S_2,S_3,S_4,S_5$ have been tested as a middle point and the last performed test $S_4S_5S_2$ has been a successful forward test. However, $S_2$ is not a convex hull point and hence the proposition is incorrect.

The next tests in our example are:

$S_5S_2S_3$, $S_4S_5S_3$, $S_5S_3S_4$.

The remaining points $S_3,S_4$ and $S_5$ form true convex hull because of successful tests $S_3S_4S_5$, $S_4S_5S_3$ and $S_5S_3S_4$.

We give a correct form of above proposition.

PROPOSITION. If all points in CH(S) have been tested as a middle point and there is a point tested twice so that all remaining points in CH(S) have been tested between these two successful testes then the algorithm is complete.

Proof. Let $S_1,S_2,\ldots,S_n$ be the remaining points in CH(S). Assume that $S_1,S_2,\ldots,S_n$ have been tested as middle points and the next test is successful test for $S_1$. This implies that there is a point $S'$ (not necessarily on the convex hull) such that tests $S'S_1S_2$, $S_1S_2S_3$, $\ldots,S_{n-1}S_nS_1$ and $S_nS_1S_2$ have been successful. Thus CH(S) consists of only the extreme points of S.

From the proof it follows that the proposition in [2] is correct if the original point is choosen as a convex hull point (this restriction is not mentioned in [2]).

As a consequence of their proposition Koplowitz and Jouppi have shown that 2N - M + 1

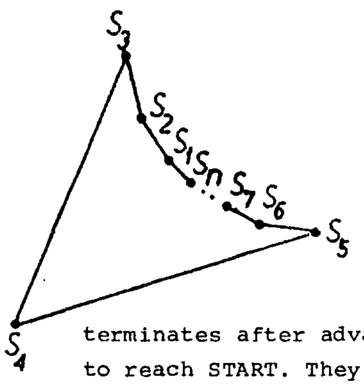


Fig. 2

tests are sufficient where M is the number of points in CH(S). In our example N = 5.M = 3 and we need 2N - M + 2 = 9 tests. Thus 2N - M + 1 tests will not assure completition of the algorithm unless the starting point $S_1$ belongs to the convex hull.

Let us consider the exmple on Fig. 2. We have the following tests:

$S_1S_2S_3$, $S_nS_1S_3$, $S_{n-1}S_nS_3$, $S_{n-2}S_{n-1}S_3$,... ...,$S_5S_6S_3$,$S_4S_5S_3$, $S_5S_3S_4$,$S_3S_4S_5$.

There are exactly N tests in this case. This contradict the following assertion in [2].

"The minimal number of tests is N and occurs when no points are deleted. For each deleted point an extra test is necessary. This N - M extra tests are made making a total of 2N - M tests".

The correct assertion is that for each deletedpoint no more than one extra test is necessary. From this it follows that the number of necessary tests is between N and 2N - M + 2. Hence the correct stopping rule is not as efficient as it seems in [2]. We described a more efficient stopping rule. If the points are sorted after one of the coordinates then we have a convex hull vertex for free. Thus, it is not of great interest to design a stopping criterion that depends on an arbitrary starting point because it is so trivial to start on a convex hull vertex. However Graham´s algorithm sorts points in terms of increasing angle according to an interior point and finding a min or max requires additional time. Also, it is of interest in case where the convex hull of starshaped polygon (the polygon with all vertices visible visible from an interior point) is to be obtained.

We will also discuss the modification given in [3,4]. Preparata, Lee and Shamos arrange the sorted points into a doubly-linked circular list with vertex labeled START (for instance, the rightmost smallest-ordinate point) as an initial vertex. The NEXT and PRED are associated with a node pointing respectively to its successor and predecessor in the list. The scan terminates after advancing all the way around to reach START. They propose the following Graham´s scan algorithm.

```
BEGIN
     v:= START;
     WHILE(NEXT[v] ≠ START DO
          IF the test(v,NEXT[v],NEXT
          (NEXT[v])) is successful
               THEN v:= NEXT[v]
               ELSE  BEGIN delete NEXT[v];
                     v:= PRED[v]
                     END
END.
```

While they describe a correct rule, their code is incorrect. As an counterexample we consider the points on Fig. 1 with START pointed to $S_5$. The first test $S_5S_1S_2$ is unsuccessful and after deleting $S_1$,v points to $S_4$ and NEXT[v] to $S_5$, i.e. to START, and the algorithm stops immediately. This happens because the stopping rule (the while condition) is incorrect. We presented a correct version of the algorithm that works for any choice of the initial point START. The stopping rule reflects the proposition proved in the section.

REFERENCES

[1] R.L.Graham, An efficient algorithm for determining the convex hull of a finite planar set, Inform.Process.Lett., 1(1972),132-133.

[2] J,Koplowitz and D.Jouppi, A more efficient convex hull algorithm, Inform.process.Lett., 7(1978),56-57.

[3] D.T.Lee and F.P.Preparata, Computational geometry-a survey, IEEE Trans.Comput., C-33(12)(1984),1072-1101.

[4] F.P.Preparata and M.I.Shamos, Computational Geometry-an Introduction (Springer-Verlag New York Inc., 1985).

# COMPUTER PROGRAM FOR DETERMINING AN OPTIMUM SOLUTION IN LONG-TERM FOREST EXPLOITATION PROCESS

INFORMATICA 4/1988

UDK 519.854/.857

Janez Barle, Janez Grad
Ekonomska fakulteta Borisa Kidriča, Ljubljana

One can define the long-term forest exploitation process in different ways. Methods of operations research can be used successfully for this purpose. A group of experts at the University of Ljubljana, Yugoslavia developed a specific tree-growing function and, based on that, prescribed a set of procedures, relations and states which could represent mathematical model for optimizing the forest exploitation process. The mathod was defined as a descrete dynamic programming process, based upon Bellman's principle. In this paper we briefly describe the model and computer program prototype for solving it. They both can easily be extended by introducing more complexity into them. The program was written in FORTRAN and tested on DEC-10 computer.

## 1. INTRODUCTION

In the paper we describe a possible usage of dynamic programming, based upon Bellman's principle, in order to obtain the optimum solution of a predefined long-term forest exploitation process. In this process measures and activities are introduced at each step within the iterative solution process and their effects on the intermediate results and the final solution are stated in the form of mathematical relations. The whole forest area which is taken into consideration is divided into a number of homogeneous parts - segments. For each of these segments, five special functions are imposed by means of which we direct (quide) the process within a number of time intervals, either years or decades. The functions are: (F1) the maximum possible tree growth capacity, (F2) exploitation capacity, (F3) the quality of the existing tree specimens, (F4) level of administration - care for improved growing conditiond, and (F5) stage in segment development, based upon the age (oldness) of the tree specimens in it. The functions help to optimize exploitation endeavours. At each step of the interative solution process four possible activities can be imposed: (A1) no activity at all, (A2) exchange of the existing tree specimen with a new one, (A3) rarefying, and (A4) restoration. The exploitation policy is defined by a sequence of chosen activities during the iterative process. The stated functions, activities and time intervals are part of the mathematical model and dynamic programming process respectively. The problem and the corresponding computer program can be extended by incorporating some additional functions and activities. Program prototype in FORTRAN was written and tested on DEC-10 computer at the University of Ljubljana, Yugoslavia. In the paper we also comment the problems which are associated with dynamic programming applied to such type of problem.

## 2. A PREDEFINED LONG-TERM FOREST EXPLOITATION PROCESS

We mean by that a stated sequence of policies and activities that should be pursued in forest exploitation within a life-time long period of some tree specimen in order to achieve optimum results in accordance with predefined goals and criterions. The possible and necessary actions and the most suitable time for these actions to be carried out depend on the values of some chosen and defined functions which describe sufficiently the tree-growing process. It is these functions, which must be predefined by a group of experts in forest exploitation process - both practicians and theoricians, by means of which more or less complexity and reality is introduced in the basic model that is to be solved. After several years of analytic and experimental studies of a group of experts in Ljubljana, Yugoslavia (VADNAL, KOTAR, ZADNIK, STIRN, GAŠPERŠIČ /2/, VADNAL, ZADNIK, STIRN /3/), the following suggestions and basic ideas have been made:

(i) The paradigm is bounded both geographicaly and in time. The whole area considered is divided into smaller regions which are further partitioned into smaller parts with specific environmental conditions and characteristics. Some parts within different regions can have similar or nearly equal environmental conditions and characteristics. They can or can not be exploited (treated) independently from each other. The area is exploited for a specific number of time periods, measured in years or decedes.

(ii) In order to evaluate (quantify) observed characteristics of each part of the forest under consideration, time dependant analytical growth functions have been developed:

1) Function of growth

$Y(t) = a(1 - (1+T)e\ x\ p(-T))$,

where $T = (2n - 1)\ t^n/(np^n)$, for $a > 0$, $p > 0$, and $n > 1$. a defines the asymptotic value of $Y(t)$, $p = t_2$ is the value of t from where on Y(t)' starts to decline, and n determines the speed of convergence Y(t) towards value a. The value of $t = t_3$, when the forest restoration procedures start can be defined by solving the equation

$1 + T + T^2 - \exp(T) = 0$

which is obtained from the relation

$t_3 Y'(t_3) = Y(t_3)$, or can be stated on the base of experiences.

2) Function of increase (by growth) $y(t) = Y(t)'$ where $y(t_2)' = y(p)' = 0$.

3) Function of average growth $f(t) = Y(t)/t$, where $f(t_3) = y(t_3)$, and therefore $t_3 Y(t_3) = Y(t_3)$.

4) Function of uniform growth $s(t) = mt$, where $s(t_3) = Y(t_3)$, and $s(t_3)' = Y(t_3)'$.

(iii) Five functions have been introduced which describe five possible states of forest exploitation process:

1) The maximum possible tree growth capacity - F1. F1 is constant during the exploitation process.

2) Exploitation capacity - F2. F2 is a continouous function, where $1 \leq F2 \leq 10$. Its value shows the degree in forest exploitation.

3) The quality of the existing tree specimens - F3.
This function enables us to make an assessment about the quality of the trees and whether to begin the exploitation process of some particular part of the forest or not. The parts are divided into five qualitative groups, $1 \leq F3 \leq 5$.

4) Level of administration - care for better growing conditions - F 4.
The value of F4, where $1 \leq F4 \leq 5$, shows a degree of obstructing influence of unwanted tree species on growth of the wanted ones.

5) The segment development stage - F5, based upon the age (oldness) of the tree specimens in it. Tree species are divided into four stages, accordingly:

$0 \leq F5 < t_1$, $t_1 \leq F5 < t_2$, $t_2 \leq F5 < t_3$, and $t_3 \leq F5 \leq t_4$*

Tree species of different stages differ both in size and quality and can therefore be used for different purposes.

(iv) The forest exploitation process is carried out by performing a sequence of prescribed activities. The sequence order of these activities helps to optimize the process. The following four activities have been taken into consideration:

1) No activity at all - A1.

In this case the forest develops in accordence with the laws of nature. The value of F2 changes only within the first two stages of segment development; it is diminished for an empirically defined quantity pr(stage, F2, A1). Similarly, the values of F3 and F4 change within the first three stages of development. They too are diminished for some empirically defined quantities pr(stage, F3, A1) and pr(stage, F4,A1), respectively. Each time the value of the increment (1 for a year or 10 for a decade) is added to the previous value of F5. At the boundary points this causes a change of the stage of development.

2) Exchange of the existing tree specimen with a new one - A2. We can pursue this activity only within the first two segment development stages. The reason for doing that is the unadequate existing tree species. The exchange is carried out there - in those parts (segments) where the effects are most visible. A2 exercises the following influence upon F2, ..., F5: Their values are changed only within the first two stages of segment development. F2, F3 and F4 are increased by some empirically defined quantities pr(stage, F2, A2),pr(stage, F3, A2) and pr(stage, F4, A2), respectively, while the value of F5 is reduced to zero and the process starts from the beginning.

3) Rerefying - A3.

The activity can only take place within the first three segment development stages. Within the first stage it is exercised by cutting down the unwanted species only what helps in quicker growth of the wanted ones. Here the activity imposes additional expenses. Within the second and third stage, in addition to the cutting down of the unwanted species, we also cut down some trees of the wanted species what helps in quicker growth of the most qualitative samples. Here the activity brings some profit. Due to A3 the values of F2, F3 and F4 are changed by some empirically obtained quantities pr(stage, F2, A3), pr(stage, F3, A3), and pr(stage, F4,A3), respectively. A3 has no particular impact on F5. The time increment (1 or 10) is added to the value of F5 after the time period expires.

4) Restoration - A4.

Restoration starts at $t = t_3$, ends at $t = t_4$, and can take different length of time. It is characterized by wood-cutting on higher scale. After restoration is done, the forest segment under consideration passes over into the first stage. The way in which the stage of restoration is being accomplished has a great influence on the results of the forest exploitation process. The span of time $t_4 - t_3$ is generally divided into more steps. At each step the increments of F2 and F3 are computed and added to the previous values of F2 and F3, respectively. F4 does not change within A4, while the increment 1 (10) is added to F5 after each year (decade) that is passing by. F5 reduces to zero or some higher value after the restoration is over.

(v) The outcomes - R(stage, Ak). Each measure - activity undertaken at any step of the exploitation process, results and can be expressed in a form of costs and income. The resulting income of some particular step, say I(stage, Ak), for k = 1, 2, 3, 4, depends upon Fi, for i = 1, 2, ..., 5, and the size of the undertaken activity Ak. The entire costs can be divided into fixed costs FIX(stage) and variable costs V(stage, Ak). Accordingly,

R(stage, Ak) = I(stage, Ak) -FIX(stage) - - V(stage, Ak)

1) R(stage, A1).

In this case we have no income and no variable costs. Therefore R(stage, A1) = - - FIX(stage)
where FIX(stage) is some empirically obtained

* where $t_4$ indicates the upper time limit of the stated tree growth cycle. $t_1$ is defined as $t_1 = a/10$.

quantity.

2) R(stage, A2).

Activity A2 is carried out only within the first two stages of the exploitation process. We have some possible income only within the second stage, which can be expressed as $q_i$(stage, A2)Y(t) where $q_i$(2,A2) is some empirically stated weight (factor), and $q_i$(1, A2) = 0. Fixed costs FIX(stage) are defined (empirically). Variable costs are obtained as a total of the costs of removal the unwanted species, say $q_c$(stage, A2)Y(t), and the costs of planting new samples, say NT(stage), where $q_c$ and NT are defined empirically. Accordingly,

R(stage, A2) = $q_i$(stage, A2)Y(t) -
-FIX(stage) - $q_c$(stage, A2)Y(t) - NT(stage)
for stage = 1,2.

3) R(stage, A3).
Activity A3 is carried out within the first three stages. The outcome R(stage, A3) can be expressed as

R(stage, A3) = $q_i$(stage, A3)Y(t) - FIX(stage) - $q_c$(stage, A3)Y(t) for stage = 1,2,3, where $q_i$, FIX, and $q_c$ are obtained empirically and $q_i$(1, A3) = 0 . $q_i$ and $q_c$ have a similar meaning as in case of R(stage, A2).

4) R(4, A4).

Activity A4 is carried out only within the fourth stage. We differentiate two possibilities:

I. $t_3 = t_4$, when all work is done in a very short time. In this case we deal with one outcome only, defined as

R(4, A4) = $q_i$(4, A4)Y($t_3$) - FIX(4) -
$q_c$(4, A4)Y($t_3$) - NT(4)

II. $t_3 \neq t_4$, and assume that there are $d_4$ steps of the exploitation process within the fourth step. At each step $n_4$, where $n_4$ = 1,2, ..., $d_4$, we compute R(4, A4, $n_4$) as

$$R(4,A4,n_4) = \frac{1}{d_4} q_i(4, A4, n_4)Y(t) -$$
$$-FIX(4) - \frac{1}{d_4} q_c(4, A4, n_4)Y(t) - NT(4, n_4)$$

where again $q_i$, $q_c$, and NT are defined empirically.

(vi) Managing the exploitation process is possible before and after the process being in progress. By this we mean some further prescribed conditions and rules which should be or should not be taken into account within the problem solving process, in accordance with the type of optimization process that we pursue. We distinguish among the following possible alternatives:

- No alternations of the prescribed exploitation process are possible while the process being in progress.
- Some alternations of the originally prescribed exploitation process are possible while the process being in progress. We may interrupt the process, insert some new input data and proceed the process from this point on or start the program from the beginning.
- The final result of the process is prescribed in advance as well as the starting conditions.
- The final result is the optimum value that can be obtained by the prescribed starting conditions.

## 3. DYNAMIC PROGRAMMING PROCESS ALGORITHM PROTOTYPE

### 3.1. General description of the descrete dynamic programming

Descrete dynamic programming process is an iterative process (BELLMAN, DREYFUS /1/). At each step i, for i = 0, 1, ..., N, we define a certain number of possible points $x_{ij}$, for j = 1,2, ..., $M_i$. For each point we compute the function value $f_{ij}$ which is involved in the process of optimization. This value is the optimum value among function values $f_{i-1\ k}$, for k = 1, 2,... $M_{i-1}$, incremented by the computed outcomes between $x_{i-1\ k}$ and this particular point $x_{ij}$. After computing $f_{ij}$ for all i = 0, 1, ..., N and j = 1, 2, ..., $M_i$ we define the optimum sequence of the operations and decisions that were made during the exploitation process for all steps i = N, N-1, ..., 0. Sometimes two or more alternatives are possible which all give the same optimum solution.

### 3.2. Forest exploitation dynamic programming algorithm

In order to start the process we need the following input data:

- a, p, n, $t_1$, $t_2$, $t_3$, $t_4$; in the prototype we don't compute the values of $t_1$, $t_2$ and $t_3$ but we read them as input data instead.

For each segment, i.e. for each case of the exploitation process:

- all empirically defined values of FIX(stage), NT(stage), $q_i$(stage, Ak), and $q_c$(stage, Ak), for stage = 1, 2, 3, 4 and k = 1, 2, 3, 4.
- all empirically defined values of pr(stage, Fi, Ak), for stage = 1, ..., 4, i = 2, 3, 4 and k =1, ..., 4, that represent the increment of Fi due to the activity Ak.
- F2, F3, F4, F5 which define the starting conditions of the forest exploitation process. The whole experiment can be repeated several times for different starting values of F2, ..., F5.

The process is as follows:

<u>I.</u>

For each t, where t = F5 + 10, F5 + 20, ..., in accordance with the stage to which t belongs (stage = 1, 2, 3 or 4), the possible activities (A1, A2, A3 and/or A4) at that stage take place for all existing (active) points $x_{t-10,s}$ (see 3.1.), where s = 1, 2, 3, ....

For each activity that takes place at some $x_{t-10,s}$ the following happens:

- new values of F2, F3, F4 and F5 are computed, defining a new point $x_{t,j}$ and a new step.

There is: new value of Fi = old value of Fi $\mp$ pr(stage, Fi, Ak), for i = 2, 3, 4, where "-" sign appears for A1, and "+" sign appers for A2, A3 or A4, respectively. The value of F5 is increased by 10 or reduced to zero (for A2 or when F5 > $t_4$).

- the outcomes R(stage, Ak), see chapter 2.(v), are computed and added to $f_{t-10,s}$ (see 3.1) in order to obtain $f_{t,j}$.

Some of the 250 possible different points $x_{tj}$, for j = 1,2, ..., 250, are encountered at each step t of the iterative process. They are defined by different values of F2, F3 and F4 (10x5x5 = 250). The computer program builds two arrays X(250, 6) and FX(250) at each step of the iterative process in order to save all the the necessary intermediate data. The elements in row j of array X contain the following data about the point $x_{t,j}$:

$x_{j1} = F2$, $x_{j2} = F3$, $x_{j3} = F4$, $x_{j4} = k$, $x_{j5} = s$, $x_{j6} = F5$

where

k (= 1, 2, 3 or 4) defínds the activity Ak taken at step t-10 that caused a transition to $x_{t,j}$, and

s definds the point $x_{t-10,s}$ from which the transition to $x_{t,j}$ was made.
s stands for the row number of array X at previous step (t-10) in which the data about $x_{t-10,s}$ are stored.

The elements $FX_j$, for j = 1, 2, ..., 250, represent the values of the computed $f_{tj}$.

II.

After completing the first part of the algorithm we locate the optimum value $f_{T,j}$ in the last step T of the interative process, where

$f_{T,j} = \underset{s}{\text{optimum}} \ (f_{T,s}$, for s = 1, 2, ..., 250)

Afterwards we trace back to the beginning all actions that were carried out during the exploitation process. We do this by means of data stored in arrays X and FX.

If the number of steps in the iterative process is fixed and defined, let say by $t_4/10$, then $T = t_4$ in case when the starting value of F5 = 0, and $T < t_4$ otherwise.

## 4. CONCLUSIONS ON THE APPROACH

Obtained experiences show that a method for optimising the forest exploitation process, based upon descrete dynamic programming is reasonable and adequate. More complexity and necessary modifications can easily be introduced after obtaining and analyzing some experimental results. Large number of input data and intermediate results which are storage demanding may be regarded as the only inconvenience. Different strategies may reduce this problem by applying the secondary disk storage at each step of the interative process. The empirically obtained input data can also be stored in files in advance and kept there for as long as necessary.

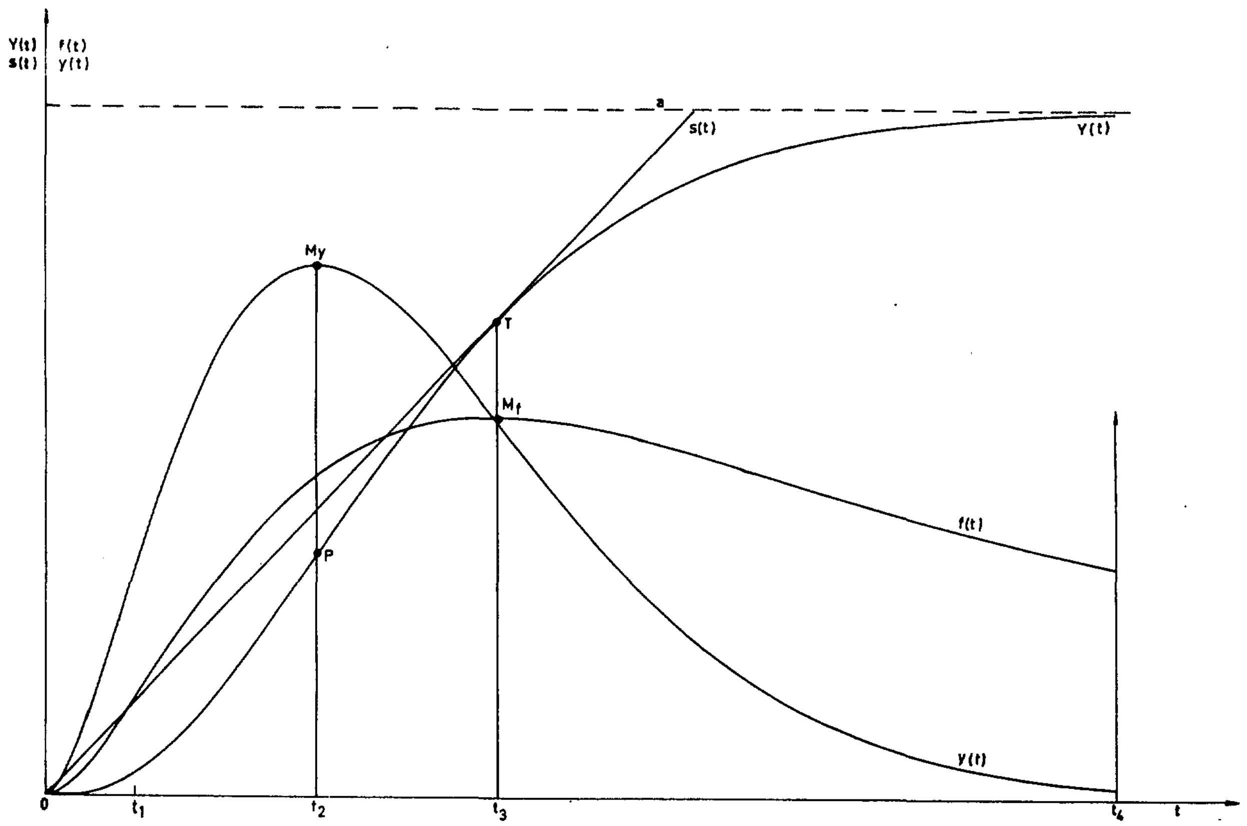
## REFERENCES

1. Bellman, R., E., Dreyfus, S., Applied Dynamic Programming, Princeton University Press, Princeton, New Yersey, 1962.

2. Vadnal, A., Kotar, M., Zadnik Stirn, L., Gašperšič, F., "Uporaba rastnih funkcij v gozdarstvu", Zbornik gozdarstva in lesarstva 23, str. 149-179, Ljubljana 1983, Yugoslavia.

3. Vadnal, A., Zadnik Stirn, L., Mathematical Model for Long-term Silvicultural and Utilization Planning, SYM OP IS'86, Herceg Novi, 7-10 October 1986, Proceedings, p.p. 183-189, FON, Belgrade 1986, Yugoslavia.

Fig.1: Analytical growth functions[2, 3]



RAČUNALNIŠKI PROGRAM ZA DOLOČITEV OPTIMALNE REŠITVE V DOLGOROČNEM IZKORIŠČANJU GOZDOV. Proces dolgoročnega izkoriščanja gozdov moremo opredeliti na več načinov. Primerne v ta namen so tudi metode operacijskega raziskovanja. Skupina strokovnjakov Univerze Edvarda Kardelja v Ljubljani je za določitev matematičnega modela optimizacije postopka izkoriščanja gozdov razvila posebno rastno funkcijo in na temelju lete definirala zaporedje potrebnih postopkov, relacij in postulatov. Metoda je bila definirana kot postopek, ki temelji na diskretnem dinamičnem programiranju. V članku zgoščeno opišemo prototipa modela in računalniškega programa za njegovo rešitev. Model je možno razširiti z vgraditvijo nadaljnjih zahtev in pogojev. Program je napisan v programskem jeziku FORTRAN in je bil testiran na računalniku DEC-10.

UDK 681.3:519.683

Radovan Andrejčič
Univerza v Mariboru
Bojan Peček
Iskra-Delta, Ljubljana

In a software life cycle the most expensive is the phase of maintenance. A lot of costs are caused by different solutions of similar problems. The programmers source code is an exhibitional example of a variety of algorithms solving the similar tasks. In larger programmer groups there are often arranged some kind of rules for programming the source code. They are nearly always called "programming standards".
135 programs were analyzed from four programmers groups (computer centers) each consisting of 2 - 4 programmers. Statistical methods such as statistical testing, sampling, analysis of variance, etc were used as an unbiased judge. Without knowing intermediate appointments about the programming was compared the source code between programmers, programmers within groups and the code between groups. Differences between programmers within groups were surprisingly small.

Vzdrževanje je najdražja faza življenskega cikla programske opreme. Zelo veliko stroškov povzroča raznolikost reševanja podobnih problemov. V večjih programerskih skupinah si oblikujejo neke vrste pravila programiranja, ki jih skoraj vedno imenujejo "programski standardi".
Delo opisuje analizo 135 programov iz štirih programerskih skupin (računalniških centrov) sestavljenih iz 2 - 4 programerjev, ki temelji na nepristranskih statističnih testih, vzorčenjih, analizi variance itd. Brez poznavanja dogovorov o programiranju je bila primerjana programska koda med programerji, programerji v okviru skupine in skupinami. Presenečajo nepričakovano majhne razlike med programerji v okviru skupin.

## 1 Introduction

Many software life cycles from different authors have been proposed. They differ in unimportant details. It is common to all of them, that the phase of maintenance is the most expensive. This phase is now the major programming activity, and very soon more programmers will be performing maintenance than development [Jones p. 35].

How to reduce costs of maintenance? Few would disagree that the quality software is not less expensive. But what is the quality software anyway?

## 2 Software Quality

It is as hard to define as defining a "good car driving". It is differently comprehended from a programmer to another programmer, from one manager to another etc. With the most known facets the software quality can be defined [by Arthur] as:

```
software quality = F(correctness, efficiency,
                     flexibility, integrity,
                     maintainability,
                     portability, reliability,
                     reusability, testability,
                     usability)
```

where each facet can be further reviewed through more criterias. For an example the maintainability can be presented as:

```
maintainability = F(concision, consistency,
                    modularity, simplicity,
                    instrumentation,
                    self-documentation)
```

Some of these criterias are easy to measure, others are not. Everyone can explain modularity, but descriptions vary from one person to another - from equalling modularity with the structured programming, over equalling with a "no GOTO programming", to a philosophy of cohesion and coupling.

And concision and simplicity? Specter of answers is nearly unlimited. Different comprehensions cause different solutions. And this is very often a reason which makes programmers spend more time and money to understand the other programmer than to solve the problem.

Achieving an uniform coding through exact standards is not realistic. "Many rules do have legitimate exceptions" [Grauer p. 92]. But on the other hand - nearly every group of programmers or computer center elaborates its own philosophy of programming. That guidelines are usually called "programming standards". So, a kind of uniformity is possible. But how much?

## 3 Source Code Analysis

### 3.1 Technics

It is of course impossible, or at least too expensive to extract data from a sample by hand. A tool or tools are needed.

Our research of the source code has based on two programs. The first one has been oriented on the analysis of the WORKING-STORAGE section. Its input has been the cross reference and the map listing produced by the compiler. Results have given information about distributions of variable descriptions, number of references, number of words in variables, paragraphs, USAGE clauses, etc.

The other program has been oriented on a procedure division. It has produced a table of usages of the COBOL reserved verbs. Occurrences of each verb have also been analyzed in the IF statement. Logical operators have been counted detail in either IF and PERFORM UNTIL statements. This program has also given a number of comments, number of paragraphs, sections, library lines of COPY statements, total number of verbs etc.

Both programs as well as the whole research were done under the DELTA/V V2.0 operating system. Because the majority of the sample programs were written for the PDP-11 computer with the DELTA/M operating system, a little recoding was sometimes needed. What does this mean for the transportability of programs? (This interesting question is not the subject of this paper).

### 3.2 Sample

#### 3.2.1 Criterion for a Sample

Collecting and analyzing a sample is not only a technical problem, but also an operational one. Very important question is immediately arisen: which programs to include in a sample - every program of an application or just the significant ones? In the first case, the analysis gives the exact answer about the application. But this perfection can hide differences between similar programs. It might show greater similarity than it really exists.

In our research the second method has been used.

#### 3.2.2 Sample Size

Four applications (programming groups) from different computer centers were included into the sample. It was common to all of them that they used the same computer language - COBOL and each group had formed some kind of its own programming rules. It is not worth mentioning that they all sweared on the structured programming (which was prescribed in their "standards").

In this paper applications are marked with letters "A" through "D" and programmers within a group with numbers. Data in table 1 have no significant meaning. They are presented just as an illustration of a sample size.

Table 1 - Illustration about the Sample Size

| Prog ramr | No p rgms | No of lines | Average lin/progr | Standar deviat. | Exec. verbs |
|---|---|---|---|---|---|
| A1 | 11 | 5274 | 479.45 | 277.0 | 2544 |
| A2 | 10 | 5517 | 551.70 | 331.6 | 3079 |
| A3 | 5 | 1344 | 268.80 | 180.0 | 706 |
| A4 | 9 | 4550 | 505.56 | 209.4 | 2908 |
| B1 | 16 | 13977 | 873.56 | 349.6 | 5116 |
| B2 | 17 | 17706 | 1041.53 | 555.2 | 6984 |
| B3 | 2 | 2548 | 1274 | 393 | 1029 |
| C1 | 15 | 10599 | 706.60 | 195.9 | 3042 |
| C2 | 10 | 6437 | 643.70 | 323.1 | 2187 |
| D1 | 22 | 45908 | 2086.73 | 617.6 | 19147 |
| D2 | 8 | 15606 | 1950.75 | 775.2 | 6834 |
| D3 | 10 | 11149 | 1114.90 | 389.3 | 3518 |
| sA | 35 | 16685 | 476.71 | 281.9 | 9237 |
| sB | 35 | 34231 | 978.03 | 475.8 | 13229 |
| sC | 25 | 17036 | 681.44 | 256.4 | 5229 |
| sD | 40 | 72663 | 1816.58 | 731.2 | 29499 |
| SUM | 135 | 140615 | 1041.59 | 731.9 | 57194 |

### 3.3 Analyzing Comment Statements

#### 3.3.1 Importance of the Comment Statements

"Although COBOL is often thought of as a self-documenting language, this is only partially true. With a careful choice of words, each statement can indeed be self-documenting, but it cannot explain its own purpose: it merely states its contribution to a technique or algorithm" [Ledin, Kudlik, Ledin p. 97].

Comments are still needed, they become even more and more important. Specially in the last time, when programs are often not maintained by the original author. As Yurdon says "No programmer, no matter how wise, how experienced, how hard pressed for time, no matter how well intentioned, should be forgiven an uncommented program".

#### 3.3.2 Number of Comments per Source Code

Absolute number of comments in a program does not have any meaning. It needs to be compared with the number of source lines, or the number of executable statements, or with the reserved COBOL verbs. Table 2 presents data about the number of source lines per comment where source lines per comment (SLC) is calculated as

$$\mathrm{SLC} = \frac{\text{total number of lines}}{\text{number of comments}}$$

Table 2 - Source Lines per Comment

| Prog ramr | No p rgms | No of comment | Average SLC | Standar deviat. | Sum of squares |
|---|---|---|---|---|---|
| A1 | 11 | 257 | 20.54 | 5.76 | 5003.5 |
| A2 | 10 | 318 | 17.35 | 3.48 | 3131.2 |
| A3 | 5 | 95 | 14.14 | 4.43 | 1097.7 |
| A4 | 9 | 223 | 20.37 | 5.90 | 4046.0 |
| B1 | 16 | 1820 | 7.68 | 2.52 | 1044.3 |
| B2 | 17 | 2720 | 6.51 | 1.54 | 761.1 |
| B3 | 2 | 432 | 5.9 | 1.50 | 74.1 |
| C1 | 15 | 2611 | 4.06 | 0.49 | 250.8 |
| C2 | 10 | 1387 | 4.64 | 1.63 | 241.7 |
| D1 | 22 | 8140 | 5.64 | 0.42 | 702.8 |
| D2 | 8 | 2922 | 5.34 | 1.22 | 239.9 |
| D3 | 10 | 2418 | 4.61 | 0.5 | 215.1 |

### 3.3.3 Differences in Percentage of Comments between Applications

Is there any significant stability in commenting programs? This answer was researched with the analysis of variance. In table 3 there is an analysis of not only differences between programs and programmers, but also of differences between applications. [Andrejcic p. 161].

Table 3 - Analysis of Variance between Applications and Programmers

| Source of Variation | Deg fre | Sum of square | Mean square | Calculated F | F table |
|---|---|---|---|---|---|
| Applications | 3 | 4506.4 | 1502.1 | 37.33 | 23.70 |
| Programmers | 8 | 207.3 | 25.91 | 2.92 | 2.663 |
| Programs | 123 | 1091.2 | 8.87 | | |

The first null hypothesis - that differences between groups (applications) are not significant can be absolutely rejected ($F_{(3;0.001)} = 23.7$). The second null hypothesis, that differences between programmers do not exist can be rejected too, but the risk is this time for a bit greater - over half a per cent ($F_{(8;0.01)} = 2.663$). This has given a reason for a detailed investigation about differences between programmers within groups. (See table 4).

Table 4 - Analysis of Variance between Programmers within Applications

| Source of var. | Applic | Deg fre | sum squar | mean squ. | calculated F | F(0.1) table |
|---|---|---|---|---|---|---|
| between | A | 3 | 184.3 | 61.4 | 2.125 | 2.28 |
| prog | B | 2 | 13.7 | 6.9 | 1.502 | 2.49 |
| ramm | C | 1 | 2.0 | 2.0 | 1.549 | 2.88 |
| ers | D | 2 | 3.62 | 3.6 | 7.262 | 2.44 |
| between | A | 31 | 896.2 | 28.9 | | |
| prog | B | 32 | 146.6 | 4.6 | | |
| ramms | C | 23 | 30.0 | 1.3 | | |
| | D | 37 | 18.5 | 0.5 | | |

These analyses have shown, that the only application in which significant differences exist was the application "D" (for the risk of 10%, but it is not greater for the risk of 0.1% - $F_{(2;0.001)} = 7.29$). The t-test proved that the programmer "D3" had more comments than the other two. Results have shown greater stability in writing comments than it had been expected.

Comparing the average (9.03 source lines per comment) of this sample with the previous investigation gives also unexpected results. Al-Jarrah-Torsun (page 344) have counted an average of 66.6 source cards per comment card. It is such a great difference, that it needs no special statistical prof. It does not also need the result of Smolej-Korelic - 23.82 lines per comment.

### 3.3.4 Correlation between Program Length and Number of Comments

Naturally, it is expected that longer programs are more complex and for this reason they need to be more commented. But the previous investigation of dr. Smolej and Korelic(*) discovered unexpected negative correlation between comments and characteristics of complex programs.

Table 5 - Correlation between Program Length and Density of Comments

| Prog | Coe.cor. | t | t(table) |
|---|---|---|---|
| A1 | 0.160 | 0.486 | t(0.50; 9) = 0.703 |
| A2 | -0.186 | 0.525 | t(0.50; 8) = 0.706 |
| A3 | -0.702 | 1.707 | t(0.10; 3) = 2.353 |
| A4 | 0.771 | 3.199 | t(0.01; 7) = 3.499 |
| B1 | -0.186 | 0.708 | t(0.40;14) = 0.868 |
| B2 | 0.148 | 0.580 | t(0.50;15) = 0.691 |
| B3 | 1.000 | | |
| C1 | 0.277 | 1.038 | t(0.20;13) = 1.350 |
| C2 | -0.535 | 1.790 | t(0.10; 8) = 1.860 |
| D1 | 0.304 | 1.428 | t(0.10;20) = 1.725 |
| D2 | 0.705 | 2.435 | t(0.05; 6) = 2.447 |
| D3 | 0.409 | 1.269 | t(0.20; 8) = 1.397 |

According to table 5 programmer "A4" is the only one who can be assumed to have larger programs less commented. The risk of rejecting the null hypothesis, that the correlation coefficient is not significant, is about over 1%. The nearest result of the programmer "D2" increases this risk up to over 5%.

Four programmers ("A2", "A3", "B1" and "C2") had even negative coefficient correlation. This means that the larger programs had relatively more comments. Programmer "A3" had this coefficient even -0.702, but his amount of the sample (5) was too small to reject the null hypothesis. Programmer "C2" with the coefficient correlation -0.535 and greater amount of sample (10) was much closer to the rejection of the null hypothesis.

### 3.3.5 Sampling Contents Of Comments

"The mere presence of comments, however, does not ensure a well-documented program, and poor comments are sometimes worse than no comments at all" [Grauer p. 103].

There are also known the first rules which suggest how to write comments (to explain reason, not to duplicate code, etc). Their present usage can be compared with the first considerations about structured programming in the early '70.

How to establish the quality of comments? At least two problems occur. The first one is to distinguish good comments from bad ones. It is impossible to do it automatically. A man as an observer and arbiter is needed. And this causes the second problem. The amount of comments is too large to examine every comment line.

-----

* Dr. Smolej and Korelic have analyzed 238 programs written by 8 programmers from one computer center with the goal to find representative characteristics of an average program.

Sampling was chosen to give an illustration of the quality of comments. Samples of comments were collected from each programmer. Each line was subjectively estimated as good or bad. Criterias for a good comment line were easy to satisfy. Each line that might be of any help in understanding the code was accepted as good. No additional comparing with the nearest lines of source code was done. Also the AQL was very low - 10% with the risk of 5%. MIL. STD. 105D double sampling plan was used.

Table 6 - Sampling the Quality of Comments

| Prog ramr | Numb. comm. | First sample amo. | AC | RE | res | Second sample amo. | AC | RE | res | Acc Rej |
|---|---|---|---|---|---|---|---|---|---|---|
| A1 | 257 | 20 | 3 | 7 | 4 | 20 | 8 | 9 | 7 | AC |
| A2 | 318 | 32 | 5 | 9 | 14 | | | | | RE |
| A3 | 95 | 13 | 2 | 5 | 4 | 13 | 6 | 7 | 10 | RE |
| A4 | 223 | 20 | 3 | 7 | 12 | | | | | RE |
| B1 | 1921 | 80 | 11 | 16 | 46 | | | | | RE |
| B2 | 2720 | 80 | 11 | 16 | 39 | | | | | RE |
| B3 | 432 | 32 | 5 | 9 | 17 | | | | | RE |
| C1 | 2610 | 80 | 11 | 16 | 5 | | | | | AC |
| C2 | 1378 | 80 | 11 | 16 | 7 | | | | | AC |
| D1 | 8140 | 80 | 11 | 16 | 3 | | | | | AC |
| D2 | 2923 | 80 | 11 | 16 | 7 | | | | | AC |
| D2 | 2418 | 80 | 11 | 16 | 6 | | | | | AC |

Results have very clearly rejected programmers with the less commented programs and accepted programmers with better results. What coincidence? Obviously, some groups take a great care about this problem, while the others do not!

## 3.4 Analyzing User-Defined Words

Beside correct comments a mnemonical significant data names are very important for understanding a data flow. Nearly all authors who deal with programming techniques suggest to use as many of the 30 characters as needed to make names in a program easy to understand. Not only to the original author, but for others as well.

Maybe this is a reason for a surprise when Al-Jarrah-Torsun discovered that the average user defined name had "only" 7.81 characters. In the next table distributions of all user-defined names are shown. Results are grouped into classes of three lengths. The hyphen is counted as the other characters.

Only programmers "B1" and "B2" have user-defined names longer than 18 characters.

In applications "C" and "D" similarities are immediately seen. Not only tests of the mean values and F-tests of the intermediate differences of standard deviation, but also much harder the Kolmogorov-Smirnov test of goodness of fit have proved that there were no differences in distributions between programmers within groups. This was specially surprising in the application "D", where distribution of each programmer was bimodal. (Every programmer had more variables with length of 13, 14 or 15 characters than with 10, 11 or 12). The first explanation was, that this was caused by the influence of the "COPY" statements. But further analyses had contradicted this suspicion.

The Kruskal-Wallis procedure [Andrejcic p. 348]

$$H = \frac{12}{12 * (12 + 1)} * \left(\frac{1369}{4} + \frac{100}{3} + \frac{361}{2} + \frac{144}{3}\right) - 3 * (12 + 1) = 7.47$$

compared with chi-square $X_{(0.05;3)} = 7.81$ gave no reason to reject the null hypothesis, that there were no significant differences between applications. However, the result was very near to the border value for the risk of 5%.

## 4 Conclusion

### 4.1 Interpretation

The first discovery was that the relative number of comments is increasing (comparing with the oldest analysis by Al-Jarrah-Torsun and a bit younger by Smolej-Korelic). All applications were produced with the interactive editor, while Al-Jarrah-Torsun wrote about cards. So, maybe also the economical effects can have some influence on the density of comments.

Not only the density, but also the constancy was surprising. It was even not effected by the program length, as it had been measured by the previous analysis. Influence of the group agreements on the programmer were reflected immediately. This brings to a conclusion, that commenting is given more and more care. It has now its place also in "the programming standards".

Table 7 - Distributions of Lengths of the User-Defined Names

| Length | A1 | A2 | A3 | A4 | B1 | B2 | B3 | C1 | C2 | D1 | D2 | D3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1-3 | 100 | 66 | 95 | 248 | 96 | 19 | 15 | 384 | 272 | 180 | 59 | 30 |
| 4-6 | 334 | 548 | 105 | 333 | 596 | 995 | 308 | 1643 | 1026 | 2168 | 813 | 731 |
| 7-9 | 419 | 270 | 82 | 323 | 1070 | 1400 | 266 | 341 | 221 | 3966 | 1269 | 1072 |
| 10-12 | 72 | 153 | 4 | | 315 | 760 | 58 | 76 | 76 | 385 | 238 | 179 |
| 13-15 | 22 | 50 | | | 255 | 301 | 9 | 30 | 10 | 1155 | 376 | 296 |
| 16-18 | | 11 | | | 186 | 126 | 2 | 2 | | 21 | 16 | 13 |
| 19-21 | | | | | 142 | 38 | | | | | | |
| 22-24 | | | | | 105 | 1 | | | | | | |
| 25-27 | | | | | 94 | | | | | | | |
| 28-30 | | | | | 44 | | | | | | | |
| average | 6.6 | 7.2 | 4.9 | 5.2 | 10.7 | 8.7 | 6.9 | 5.6 | 5.5 | 7.9 | 8.0 | 8.2 |
| rank | 8 | 6 | 12 | 11 | 1 | 2 | 7 | 9 | 10 | 5 | 4 | 3 |

Sampling of comments gave some disappointing results, or at least unexpected. It was very easy to distinguish between the good and the bad comments. Criterias were easy to achieve, but results rejected programmers with the less commented programs.

After the research was finished, each "programming standard" was studied in detail. Results of the analysis were compared with these prescriptions. In applications"C" and "D" detailed programming guidelines about the form of comment were stated, while in others they were omitted.

Al-Jarrah-Torsun found that the average user-defined name had 7.81 characters and their expectation that it "was expected to find them to be on average much longer" [Al-Jarrah-Torsun p. 343] was not in place. It seems that an average of 8 characters is the most common value. Equality of distributions of the user-defined names was greater than expected. Descriptional estimates about the importance of the long user-names were:

B1 - very important
B2 - very important
B3 - very important
C1 - less important
C2 - less important
D1 - very important
D2 - important
D3 - important

An interview with programmers on the application "A" was not possible. Answers were as expected, except the programmer B3's and D1's. Programmer "B3" was a beginner and the worst typist. "D3" was also very bad, the worst in his group, but they both answered under impression of the group agreements. If the programmer "B3" would be separated, the Kruskal-Wallis procedure

$$H = \frac{12}{11 * (11 + 1)} * \left(\frac{1156}{4} + \frac{9}{2} + \frac{289}{2} + \frac{144}{3}\right) - 3 * (11 + 1) = 8.18$$

would reject the null hypothesis (with the risk of 5%), that there were no differences between applications about lengths of the user names. This would prove, that the statistical significant differences exist. For this reason the correlation between the typing speed and the length of user-names was not analyzed. As there were nearly no differences between programmers within groups, results were obviously more depended on agreements than the dexterity. The suspicion, that the uniformity of distribution was caused by the COPY statements in the WORKING-STORAGE section was comprehended. The amount of user-names from the library files was found to be very low.

With the method of comparing the mean value with the constant it was evidenced that each programmer had different average than the sample of Al-Jarrah-Torsun (7.81) with the greatest risk of 3.67 for the programmer "D2". Coincidently, the whole sample together had an average of 7.79 with standard deviation of 3.62, so critical risk (CR) [Andrejcic p. 100]

$$CR = \frac{7.81 - 7.79}{\frac{3.62}{\sqrt{27484}}} = 0.916$$

gave no reason to contradict the hypothesis that the both samples had statistically equal mean values.

## 4.2 Comment of the Analysis

It needs to be stated clearly, that the goal of this analysis was not to point to the quality of the software. The goal was to find similarities and differences between applications and programs within an application. And this paper is only to give a short illustration of the analysis, so only a part of the research is shown. There are of course more calculations and comparisons.

This analysis neither measures nor estimates the quality of applications. It is impossible to do it just on some facets about the state of the source code. It is well known, that the quality of the software is designed and determinated in the previous phases of the software life cycle.

The quality of the source code is not the most important component of the software quality. So, it cannot be made equally with the software quality which make part of the linear equation [by ROLAND]

$$X = W_1 * X_1 + W_2 * X_2 + \ldots + W_n * X_n + C$$

where W's are weighting factors and X's are software metrics - each of which may be or may not be given in turn by linear equations of the same form, and C is the constant. One of them is also the maintainability as it had been shown at the beginning. Uniformity of code can be of a great help in eliminating difficulties and frustrations in authorship of the program.

Anyhow, the analysis proved that differences within groups were very small, but "programming standards" were different. Even if they all referred to the same philosophy, they are a great reason for different solutions of similar problems. And our opinion is that this is a subjective argument, which needs to be eliminated. Our analysis examines for the realistic possibilities to achieve it.

## References

1. dr. Radovan Andrejcic: "STATISTIKA PRI KADROVANJU IN IZOBRAZEVANJU" - VSOD Kranj 1979
2. Lowell Jay Arthur: "MEASURING PROGRAMMER PRODUCTIVITY AND SOFTWARE QUALITY" - John Willy & sons 1985
3. dr. Robert T. Grauer: "STRUCTURED METHODS THROUGH COBOL" - Prentice Hall
4. Capers Jones: "PROGRAMMING PRODUCTIVITY" - McGraw-Hill 1986
5. M. M. Al-Jarrah and I. S. Torsun: "EMPIRICAL ANALYSIS OF COBOL PROGRAMS" - Software Practice and Experience 9/1979
6. George Ledin Jr., Michael Kudlick, Victor Ledin: "THE COBOL PROGRAMMER'S BOOK OF RULES" - Belmont California
7. Dr. Vitomir Smolej and Igor Korelic: "EMPIRICNA ANALIZA COBOLSKIH PROGRAMOV" - Informatica, Ljubljana oct. 1981
8. John Roland: "SOFTWARE METRICS" - Computer Lang. 6/1986
9. dr. Francis J. Wall: "STATISTICAL DATA ANALYSIS HANDBOOK" McGraw-Hill 1986

# PREVODJENJE PRAVILA GRAMATIKE IZ EBNF ZAPISA U BNF ZAPIS

UDK 519.682.1

Mirjana Ivanović, Zoran Budimac
Institut za matematiku

**TRANSLATION OF GRAMMAR RULE EBNF DESCRIPTIONS TO BNF ONES:** The use of computers in grammar processing of any kind demands the simplest possible description of grammar rules. Backus Normal Form (BNF) is acceptable solution.
However, Extended Beckus Normal Form (EBNF) is more suitable for humans. For purposes of efficient computer processing of grammar rules described by EBNF, preprocessor becomes a must. It translates EBNF description of grammar rules to equivalent BNF description.

SAŽETAK: Obrada gramatika (programskih jezika) na računaru zahteva što jednostavniji način zapisivanja njihovih pravila. Backusova normalna forma (BNF) je jedno od prihvatljivih rešenja.
Medjutim, za čoveka koji zapisuje pravila pogodnija je proširena Backusova normalna forma (EBNF).
Da bi se pomirila ova dva protivrečna zahteva realizovan je predprocesor koji obavlja prevodjenje pravila iz EBNF u BNF zapis.

## 1. Uvod

Većina programskih jezika se može, skoro u potpunosti, definisati pravilima kontekstno slobodnih gramatika, koja su oblika:

$$\alpha ::= \beta$$

Leva strana pravila je jedan od simbola iz skupa neterminalnih simbola gramatike, a desna strana pravila je konačan niz terminalnih i/ili neterminalnih simbola.

Potreba preciznog formalnog definisanja programskih jezika uslovila je pojavu različitih notacija za zapis pravila gramatike, od kojih se najviše koriste:

a) Bekusova (BNF) i proširena Bekusova (EBNF) notacija,
b) obrnuta i modifikovana obrnuta notacija,
c) grafička notacija,
d) sintaksni dijagrami.

U slučaju a) i b) sintaksa jezika se definiše pomoću konačnog skupa metaformula.

Metaformula se sastoji od leve i desne strane razdvojene univerzalnim metasimbolom ::=. Leva strana metaformule je metapromenljiva, a desna strana je metaizraz - niz metapromenljivih i/ili metakonstanti razdvojenih univerzalnim metasimbolom |.

Metasimbol | povezuje različite desne strane pravila sa istim levim stranama:

$$\begin{array}{l} \alpha ::= \beta_1 \\ \alpha ::= \beta_2 \\ \text{--------} \\ \alpha ::= \beta_n \end{array} \quad \Leftrightarrow \quad \alpha ::= \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

## 2. Primena BNF i EBNF u zapisivanju pravila gramatike

Da bi se omogućila jednoznačnost pri definisanju simbola gramatike uvode se ograničavači za obe kategorije simbola. Neterminalni simboli se ograničavaju znakom < (na početku) i > (na kraju), a terminalni znakom '. Ovakav način obeležavanja stvara dodatnu opreznost i napor pri zapisivanju pravila, ali daje veće slobode kod imenovanja simbola.

Za zapisivanje pravila kontekstno slobodnih gramatika, BNF nudi sledeću pogodnost: ukoliko postoji više pravila sa različitim desnim, a jednakim levim stranama, tada se ona zapisuju kao jedno pravilo čija je leva strana jednaka levoj strani niza pravila, a desna strana sadrži sve desne strane niza pravila razdvojene metasimbolom | (u značenju ili).

Ovakav način zapisivanja je nefleksibilan i dovodi do dupliranja i nepotrebnog ponavljanja pojedinih nizova simbola u pravilima.

EBNF donosi značajna proširenja jer pored metasimbola koje koristi BNF (::= i |) uvodi parove metasimbola: (), [], {}.

## 2.1 Metasimboli () (Grupisanje)

Desne strane pravila sa istim levim stranama se nekad razlikuju samo po nekom nizu terminalnih i/ili neterminalnih simbola. Pomoću metasimbola () uvodi se grupisanje sa sledećim značenjem:

BNF EBNF

$P ::= i_1\ r_1\ i_2$

$P ::= i_1\ r_2\ i_2$

-------------- <=> $P ::= i_1\ (\ r_1\ | \ldots |\ r_n\ )\ i_2$

$P ::= i_1\ r_n\ i_2$

Bar jedan niz simbola $i_1$ ili $i_2$ nije prazan. Takodje, $r_1, \ldots, r_n$ (za $n>1$) nisu prazni nizovi simbola.

## 2.2 Metasimboli [] (jednostruka pojava)

Dva pravila gramatike se mogu razlikovati samo po pojavi nekog niza simbola u jednom od njih. Da bi se izbeglo pisanje oba pravila uvode se metasimboli [] sa sledećim značenjem:

BNF EBNF

$P ::= n_1\ n\ n_2$

$P ::= n_1\ n_2$ <=> $P ::= n_1\ [\ n\ ]\ n_2$

Niz terminalnih i/ili neterminalnih simbola ograničen metasimbolima [] javlja se u pravilu najviše jedanput.

Da bi pravilo imalo smisla niz simbola n nije prazan.

## 2.3 Metasimboli {} (višestruka pojava)

Pravila, u kojima se neki niz simbola gramatike javlja proizvoljan broj puta, se u BNF ne mogu direktno predstaviti već se koristi rekurzija. Rekurzija se otklanja uvodjenjem metasimbola {} sa sledećim značenjem:

BNF EBNF

$P ::= n_1\ n_3$

$P ::= n_1\ n_2\ n_3$

$P ::= n_1\ n_2\ n_2\ n_3$ <=> $P ::= n_1\ \{\ n_2\ \}\ n_3$

--------------------

$P ::= n_1\ n_2 \ldots n_2\ n_3$

Da bi pravilo imalo smisla niz simbola $n_2$ nije prazan.

Moguće su sledeće pojave parova metasimbola [] i {}:

1) [...{...}...]
2) {...[...]...}
3) {...{...}...}
4) [...[...]...]

do proizvoljnog nivoa dubine. Pojava metasimbola () je moguća na svim nivoima.

## 3. Prevodjenje pravila iz EBNF zapisa u BNF zapis

Razvijanje prevodilaca za više programske jezike je jedna od značajnih oblasti računarskih nauka, stoga je potrebno na pogodan način izraziti sintaksna pravila jezika korišćenjem gramatika.

Za obradu gramatika na računaru pogodniji je zapis pravila u BNF notaciji. Sa stanovišta čoveka, jednostavnije i prirodnije je pravila zapisivati u EBNF notaciji.

Pri ručnom prevodjenju pravila uočavaju se odredjene zakonitosti koje je moguće lako programski simulirati.

### 3.1. Zamena metasimbola () (grupisanja)

Deo pravila ograničen metasimbolima ( i ), se može zameniti uvodjenjem novog neterminalnog simbola. Za uvedeni neterminalni simbol (koji je različit od svih neterminalnih simbola gramatike) definiše se novo pravilo:

$P ::= d_1\ (\ p_1\ |\ p_2\ | \ldots |\ p_n\ )\ d_2$

<=>

$P ::= d_1$ <zamena> $d_2$

<zamena>::= $p_1\ |\ p_2\ | \ldots |\ p_n$,

gde su P i <zamena> neterminalni simboli gramatike, $d_1$ i $d_2$ su nizovi simbola od kojih je bar jedan neprazan.

### 3.2. Zamena metasimbola [] (jednostruke pojave)

Od početnog pravila se formiraju dva nova:

- početno pravilo sa jednom pojavom niza terminalnih i/ili neterminalnih simbola ograničenih metasimbolima [ i ],

- početno pravilo bez niza terminalnih i/ili neterminalnih simbola ograničenih metasimbolima [ i ].

```
P ::= p1 [ n ] p2     <=>    P ::= p1 p2
                              P ::= p1 n p2
```

### 3.3. Zamena metasimbola {} (višestruke pojave)

Zamena niza terminalnih i/ili neterminalnih simbola ograničenih metasimbolima {} se može realizovati na više načina. Medutim, u većini slučajeva zamena je uglavnom uslovljena tipom pravila i ne predstavlja univerzalno rešenje.

Analizirajući nedostatke takvih zamena i samu definiciju višestruke pojave, dolazi se do opšte metode za zamenu višestruke pojave koja se može jednostavno realizovati.

Od izvornog pravila formiraju se četiri nova:

- početno pravilo bez niza simbola ograničenog metasimbolima { i },

- u početnom pravilu se niz simbola ograničen metasimbolima { i }, zamenjuje novim neterminalnim simbolom,

- novouvedeni neterminalni simbol se definiše pomoću dva pravila

```
    P ::= n1 { n2 } n3
<=>
    P ::= n1 n2
    P ::= n1 <zamena> n2
    <zamena>::=<niz zamenjenih simbola> |
               <niz zamenjenih simbola><zamena>
```

Može se učiniti na prvi pogled da su neka pravila suvišna i da bi se mogla uvesti jednostavnija zamena. Medjutim, jednostavnija zamena bi dovela do pojave praznog pravila (tj. pravila sa levom, ali bez desne strane) koje može da oteža proces rada prevodioca.

## 4. Realizacija

Navedeni algoritmi za zamenu metasimbola EBNF su osnova na kojoj se zasniva rad predprocesora. Da bi se oni uspešno i efikasno implementirali izabrana je pogodna forma za eksternu i internu reprezentaciju pravila.

### 4.1 Eksterna reprezentacija pravila

Pravila gramatike se zapisuju u datoteku. Potrebno je prilikom njihovog zapisivanja poštovati odredjene zahteve da bi se što jednostavnije koristila u obradi. Jednoznačno se identifikuju tri kategorije simbola:

- **Neterminalni simboli** su ograničeni znakom < na početku i znakom > na kraju. Naziv simbola se može sastojati od više reči. Ukoliko su reči razdvojene sa više praznih mesta uzima se u obzir samo jedno. Praznine izmedju < i prvog znaka simbola i poslednjeg znaka simbola i > se ne uključuju u simbol tj. nisu od značaja.

- **Terminalni simboli** su ograničeni znakom ' i na početku i na kraju. Niz znakova koji predstavlja simbol se koristi u istom obliku kako je i zapisan.

- **Metasimboli** se navode bez ograničavača.

Pravila moraju da zadovolje sledeće zahteve:

- maksimalna dužina simbola zajedno sa ograničavačima je 40 znakova,

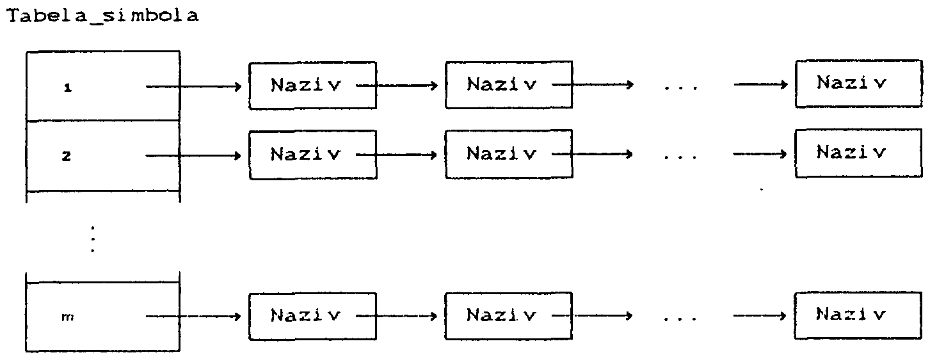
- jedno pravilo se može nastavljati u više linija,

- iza metasimbola ::= mora da sledi bar jedan simbol desne strane pravila,

- jedan simbol se ne može nastavljati u više linija,

- metasimbol | (ukoliko je potreban) stavlja se na kraj tekuće linije, a ne na početak sledeće,

- izmedju susednih simbola metajezika i susednih terminalnih simbola objekt jezika obavezna je upotreba bar jednog praznog mesta,



**Slika 4.1. Opšti izgled tabele simbola**

dok u svim ostalim slučajevima nije,

- metasimboli (,), [,], {,} se mogu ugnezditi do proizvoljne dubine,

- za svaki metasimbol (, [, { u tekućoj liniji mora da postoji njegov odgovarajući metasimbol ), ], }.

U procesu učitavanja pravila i formiranja njihove interne reprezentacije vrši se odredjena logička kontrola ispravnosti zapisa i ukoliko se otkriju neke greške (nepoštovanjem postavljenih zahteva) predprocesor zaustavlja rad i izlistava pogrešno pravilo.

## 4.2. Interna reprezentacija pravila

Za smeštanje pravila u memoriju nakon njihovog učitavanja, koristi se tabela simbola. Tabela simbola je niz od m pokazivača dinamičke strukture - liste koje sadrže slogove tipa naziv. Svaki simbol gramatike se u tabeli simbola javlja samo jednom (slika 4-1).

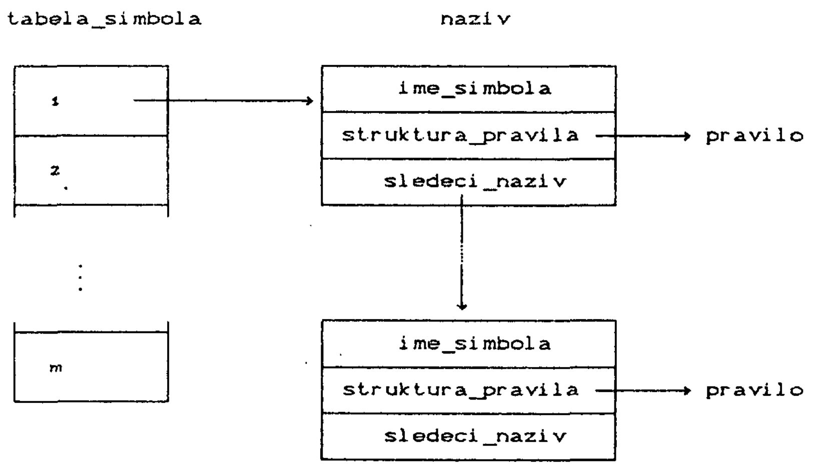
```
tabela_simbola=array[1..m] of naziv;
naziv=record
          ime_simbola:string[40];
          struktura_pravila:^pravilo;
          sledeci_naziv:^naziv;
      end;
```

Slog **naziv** se sastoji od polja:

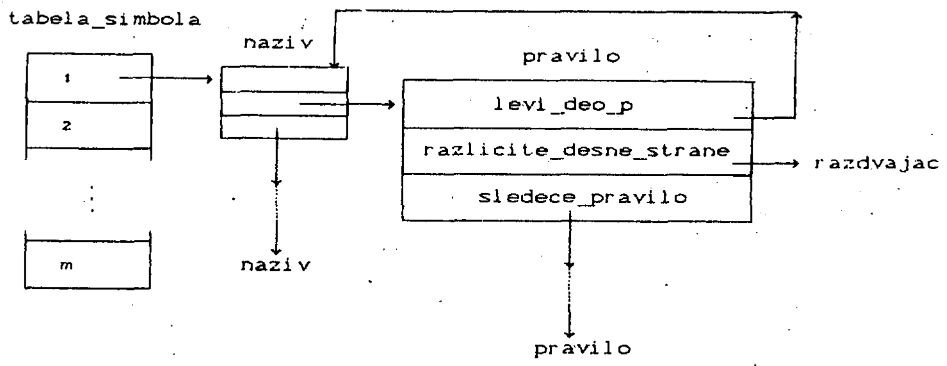
- **ime_simbola** je niska dužine 40 znakova, u koju se upisuju nazivi simbola gramatike zajedno sa ograničavačima,

- **struktura_pravila** je pokazivač na slog tipa pravilo, kojim se predstavljaju pravila za neterminalne simbole, dok je za terminalne simbole vrednost pokazivača nil,

- **sledeci_naziv** je pokazivač na sledeći slog tipa naziv u listi.



**Slika 4.2. Struktura sloga naziv**



**Slika 4.3. Struktura sloga pravilo**

Za svaki neterminalni simbol iz tabele simbola vezuje se slog tipa **pravilo** kojim se opisuju sva pravila sa istom levom stranom.

```
pravilo=record
        levi_deo_p:^naziv;
        razlicite_desne_strane:^razdvajac;
        sledece_pravilo:^pravilo;
    end;
```

Slog **pravilo** se sastoji od polja:

- **levi_deo_p** je pokazivač na naziv neterminalnog simbola (u tabeli simbola) koji se nalazi na levoj strani pravila,

- **razlicite_desne_strane** je pokazivač na slog tipa razdvajac pomoću kog se opisuju sva pravila sa istom levom stranom levi_deo_p,

- **sledece_pravilo** je pokazivač na sledeću grupu pravila.

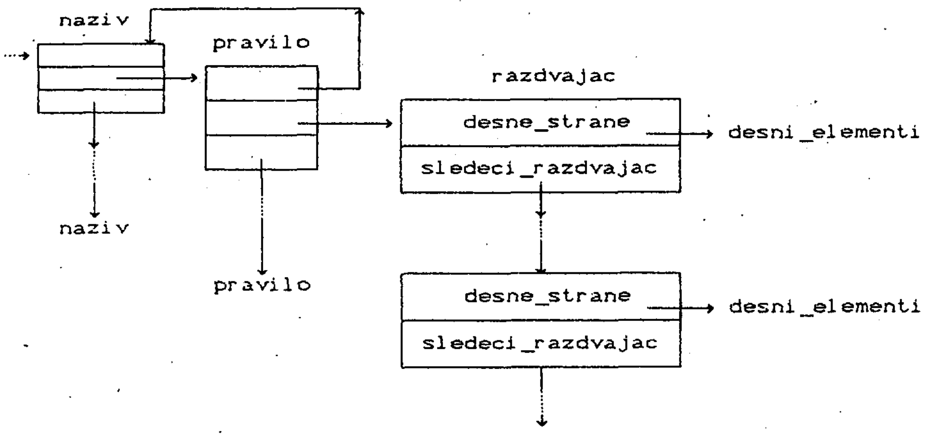
Slog tipa **razdvajac** omogućuje predstavljanje svih pravila koja pripadaju istoj grupi tj. imaju istu levu stranu,

```
razdvajac=record
        desne_strane:^desni_elementi;
        sledeci_razdvajac:^razdvajac;
    end;
```

Slog tipa **razdvajac** se sastoji od polja:

- **desne_strane** je pokazivač na listu simbola desne strane pravila predstavljenih listom elemenata tipa desni_elementi,

- **sledeci_razdvajac** je pokazivač na sledeći slog tipa razdvajac za koji se vezuje desna strana novog pravila iste grupe.



**Slika 4.4. Struktura sloga razdvajac**

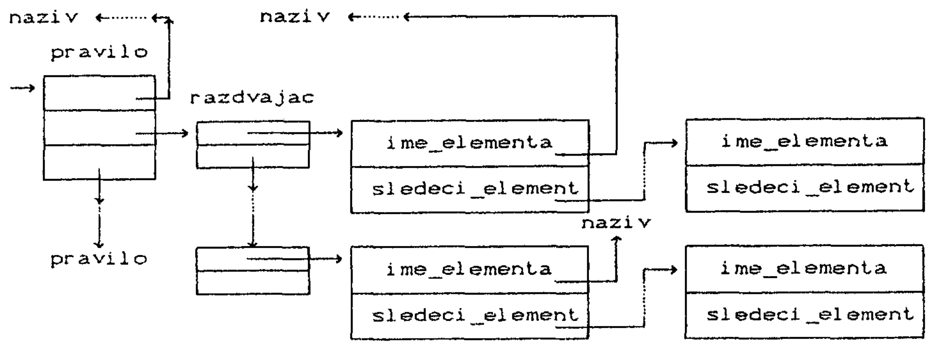
Slog tipa **desni_elementi** omogućuje predstavljanje pojedinačnih simbola desnih strana pravila.

```
desni_elementi=record
        ime_elementa:^naziv;
        sledeci_element:^desni_elementi;
        end;
```

Slog **desni_elementi** se sastoji od polja:

- **ime_elementa** je pokazivač na naziv simbola upisanog u tabelu simbola,

- **sledeci_element** je pokazivač na sledeći simbol desne strane pravila.



Slika 4.5. Struktura grupe pravila

## 4.3. Programsta implementacija

Predprocesor je realizovan u Turbo Pascal jeziku kao niz procedura i funkcija. Celokupna aktivnost predprocesora se može podeliti u nekoliko celina:

- iz ulazne datoteke učitavaju se redom linije, odvajaju se pojedinačni terminalni i neterminalni simboli, upisuju se u tabelu simbola i formira se interna struktura pravila,

- ako se u analiziranoj liniji nalazi niz simbola ograničen metasimbolima (,) taj niz se u pravilu zamenjuje novim neterminalnim simbolom, a izdvojeni niz se upisuje u listu zamena kao novo pravilo,

- *nakon-* učitavanja svih pravila iz datoteke analiziraju se izdvojena pravila u listi zamena,

- nakon toga korišćenjem algoritama zamene metasimbola [,], {,} vrši se njihova zamena uz logičku kontrolu (čuvaju se sve zamenjene niske simbola da se ne bi ista niska simbola zamenila više puta različitim neterminalnim simbolima).

Ponavljanjem ovog postupka dobija se interna reprezentacija pravila u BNF notaciji.

## 6. Zaključak

U oblasti konstruisanja prevodilaca za više programske jezike značajnu ulogu igra izbor pogodnog načina zapisa pravila gramatike jezika.

EBNF i BNF notacije su pogodan mehanizam za zapisivanje pravila i široko se koriste u svim oblastima gde se zahteva formalizovan način zapisivanja pravila.

Realizovani predprocesor kao gotov, celovit softverski proizvod realizovan je na Turbo Pascal jeziku na IBM PC kompatibilnom računaru. Može se jednostavno uključiti u šire softverske proizvode kao gotovo orudje za lakšu obradu i upotrebu pravila proizvoljne kontekstno slobodne gramatike.

## Literatura

1. Gries D. - Compiler construction for digital computers, John Wiley & sons INC., New York, 1971.

2. Ivanović Mirjana - Programska implementacija sintaksnog anaizatora programskih jezika zasnovanog na metodi prioriteta, Magistarski rad, Novi Sad, 1988.

3. Jensen K. i Wirth N. - Pascal user manual and report, Springer-Verlag, New York, 1985.

4. Tremblay J.P., Sorenson P.G. - Compiler writing, McGraw-Hill Book Company, New York, 1985.

UDK 519.688

Samo Bobek
Visoka ekonomsko-komercialna šola, Maribor

Pričujoči prispevek obravnava računalniško podprto poslovno odločanje kot sinergijsko (so)delovanje človeka in računalnika,ki ju v tem primeru povezujejo informacijski oziroma odločitveni procesi, ki se odvijajo v poslovnem sistemu. V tej vlogi naj računalnik predvsem uspešno in učinkovito podpira človekove spoznavne (kognitivne) procese in ravno tako tudi komuniciranje med posamezniki.

QUO VADIS INFORMATIZATION IN BUSINESS DECISION MAKING: This paper describes computer supported (business) decision making as cooperation (with sinergy effect) betwen humans and computers, which are in interaction through decision making process in organization. Computer should support, as part of such system, human cognitive process and communication between individuals in decision groups.

## 1 KAKSNE RAČUNALNISKO PODPRTE POSLOVNE INFORMACIJSKE SISTEME POTREBUJEMO ?

Računalniško podprte poslovne informacijske sisteme predstavljamo v smislu njihovih posebnosti in značilnosti najpogosteje skozi: (1) avtomatsko obdelavo podatkov (AOP) --> (2) upravljalni informacijski sistem (UIS) --> (3) sistem podpore odločanja (SPO). Ilustrirajmo navedene usmeritve računalniško podprtih poslovnih informacijskih sistemov, kot njihove razvojne korake, s kratkim prikazom njihovih značilnosti (glej tudi sliko 1).

Vsekakor lahko ugotovimo, da gre za vrsto kvalitativnih sprememb, ki se kažejo predvsem v (Kajzer, 1986):

- prenosu težišča s podatkov (AOP) na sporočila (UIS) in nato na informacije (SPO);
- razširitvi semiotične razsežnosi s sintakse (AOP) na semantiko (UIS) in nato na pragmatiko (SPO);
- spremembi organiziranosti podatkov od posamičnih (AOP) na bazo podatkov (UIS) ter nato na bazo modelov in podatkov (SPO);
- dopolnitvi standardnih (s)poročil z ad hoc sporočili (odgovori na vprašanja);
- prehodu od starejših postopkovnih (AOP) na sodobnejše postopkovne (UIS) in nato na nepostopkovne programske jezike (SPO);
- preoblikovanju osnove delovanja od posamičnih programov (AOP) na programske "pakete" (UIS) in nato na generatorje sistemov podpore odločanja - sisteme za reševanje problemov (SPO);
- prehodu od neintegriranosti (AOP) k integriranosti na ravni (poslovne) funkcije (UIS) in nato k integriranosti na ravni organizacije (SPO);
- povečanju stopnje prilagodljivosti od togosti do spremenljivosti (fleksibilnosti);
- razširitvi na zgodnejše faze upravljanja: od operative (AOP) na taktiko (UIS) in nadalje na strategijo (SPO);
- spremembi cilja: od težnje k učinkovitosti obdelave podatkov (AOP) k učinkovitosti (UIS) in nato uspešnosti poslovnega sistema (SPO),
- transformaciji "uporabnika" od organizacije (AOP), na posameznika (UIS) in nato na posameznika v organizaciji (SPO);
- razširitvi podpore strukturiranih odločitev na podporo semistrukturiranih odločitev in nestrukturiranih odločitev;
- prehodu iz posamičnih modelov na sisteme modelov.

Vidimo, da se je na področju poslovnih informacijskih sistemov že marsikaj spremenilo, mnogo pa se bo še moralo. V nadaljevanju se bomo omejili na vlogo računalnika kot pripomočka pri poslovnem odločanju. Pri tem bomo izhajali iz sistema človek-računalnik, kot lahko označimo (sinergijsko) sodelovanje človeka in računalnika v procesu računalniško podprtega poslovnega odločanja. Gre za še enega izmed fenomenov iz skupine, ki jih označujemo z computer aided .... (CA..) - namreč za computer aided decision making (CADM).

## 2 RAČUNALNIK KOT PRIPOMOČEK PRI POSLOVNEM ODLOČANJU

Sistem človek - računalnik sestavljajo, tako kot vsak sistem, določene sestavine, notranje povezave med sestavinami in mejne povezave z okoljem. V sistemu človek-računalnik so:

(1) sestavine izražene z množico ljudi in množico računalnikov,
(2) notranje povezave z nekim procesom, ki določa "pravila" sodelovanja ljudi in računalnikov,
(3) mejne povezave pa s podatki kot preslikavo stanja v objektivni stvarnosti ter z "rezultati", ki izhajajo iz sinergijskega (so)delovanja ljudi in računalnikov.

Sistem človek-računalnik lahko zato v kontekstu poslovnega odločanja razumemo kot (integriran) sistem ljudi in računalnikov iz katerega skozi

| Značilnosti | AOP --> | UIS --> | SPO |
|---|---|---|---|
| . težišče | PODATKI | SPOROCILA | INFORMACIJE |
| . semiotična razsežnost | SINTAKSA | SEMANTIKA | PRAGMATIKA |
| . organiz. podatkov | POSAMICNI PODATKI | BAZA PODATKOV | BAZA PODATKOV IN MODELOV |
| . sporočila | STANDARDNA (S)POROCILA | AD HOC (S)POROCILA (ODGOVORI) | |
| . programski jeziki | STAREJSI POSTOPKOVNI | NOVEJSI POSTOPKOVNI | NEPOSTOPKOVNI |
| . osnova delovanja | POSAMICNI PROGRAMI | PROGRAMSKI "PAKETI" | SISTEM ZA RESE-VANJE PROBLEMOV |
| . cilji | UCINKOVITOST OBDELAVE POD. | UCINKOVITOST ORGANIZACIJE | USPESNOST ORGANIZACIJE |
| . razsežnost upravljanja | OPERATIVA | TAKTIKA | STRATEGIJA |
| . integrira-nost | NE OBSTAJA | NA RAVNI FUNKCIJ(E) | ORGANIZACIJA KOT CELOTA |
| . stopnja prilagodlj. | TOG | OMEJENO PRILAGODLJIV | FLEKSIBILEN |
| . uporabnik | ORGANIZACIJA | POSAMEZNIK | POSAMENIK V ORGANIZACIJI |
| . vrste odločitev | STRUKTURIRANE | S E M I S T R U K T U R I R A N E | NESTRUKTURIRANE |
| . vrste modelov | P O S A M I C N I  M O D E L I | S I S T E M I  M O D E L O V | |

Slika 1: Razvojni koraki modelov poslovnih informacijskih sistemov (Kajzer, 1986).

(računalniško podprto) poslovno odločanje "izhajajo" poslovne odločitve. Proces, ki poveže ljudi in računalnike je torej v tem primeru odločitveni proces, rezultati pa so poslovne odločitve.

Če lahko računalniško nepodprto poslovno odločanje ponazorimo takole

človek --> odločitev

oziroma tudi takole, ko človek sicer uporablja računalnik, vendar ne (neposredno) pri odločanju.

računalnik --> človek --> odločitev

za razliko od računalniško programiranega odločanja, ki poteka takole,

človek --> algoritem --> računalnik --> odločitev,

potem lahko računalniško podprto odločanje ponazorimo takole

```
        človek
          ! --------> odločitev.
        računalnik .
```

```
         +-----------------------+
+------------+                   +-------------+
! človek     !  odločitveni      ! računalnik  !
!            !  proces           !             !
+------------+                   +-------------+
         +-----------------------+
```

Slika 2: Sistem človek - računalnik pri poslovnem odločanju.

Koncept sistema človek - računalnik izhaja iz predpostavke, da so nekatere naloge najlažje izvedljive, če se jih loti človek, druge pa, če se jih "loti" računalnik. V odvijanju poslovnega odločanja oblikujeta človek in računalnik (informacijska tehnika in tehnologija) sistem, iz katerega izhajajo rezultati (poslovne odločitve) in sicer skozi vrsto interakcij med računalnikom in človekom (glej sliko 2).

Izhodišča sodobnega računalniško podprtega poslovnega odločanja, ki se odvija v tovrstnem sistemu človek-računalnik, izhajajo iz naslednjih predpostavk o vlogi računalnika pri tem :

(1) Računalnik naj nudi podporo odločevalcem pri poslovnem odločanju, torej jih naj le podpira in ne nadomešča oziroma zamenjuje. Predvsem presoja ljudi je tista, ki je načeloma nikoli ne prepustimo računalniku (Keen, 1976).

(2) Poslovne odločitve naj nastajajo v neposrednem dialogu človek-računalnik, tako da računalnik nudi pomoč pri razumevanju konteksta odločanja, omogoča različne pripomočke za računalniško podporo odločitvenih procesov (operacij), zagotavlja dodatni, z vidika človeka odločevalca eksterni spomin ter dopušča učinkovito upravljanje računalnika, kot pripomočka poslovnega odločanja (Sprague, 1983).

Že v sredini 50-ih let je Robert Oppenheimer (Chorafas, 1986) ugotavljal, da je računalnik stroj, ki je popolnoma drugačen od elektronskega knjižnega stroja in od ostalih "strojev" za obdelavo podatkov. Zato ga moramo uporabljati na popolnoma drugačen način, če hočemo kar najučinkoviteje izkoristiti vse prednosti, ki nam jih nudi. Potrebovali smo tri desetletja, da smo doumeli bistvo te misli. Nadomestiti moramo zamujeno in narediti bistven korak naprej.

Računalnik moramo začeti uporabljati predvsem kot procesor "simbolov" in ne le kot procesor števil (kalkulator). Uporabimo ga za učinkovito računalniško podproro t.i. kognitivnih procesov. Le tako bo računalnik lahko resnično postal pripomoček človeku - posamezniku. To nam danes uspeva različno (ne)uspešno.

Računalnike moramo začeti uporabljati tudi za povezovanje posameznikov, ki odločajo v vzpostavljeni organizacijski strukturi poslovnega sistema. Posamezniki, ki odločajo niso le "solo šahisti", ki "igrajo" - odločajo samostojno, temveč odločajo predvsem v skupinah. Zato moramo začeti razmišljati tudi o tem, kako računalniško podpreti skupinsko odločanje (DeSanctis, 1984) ter kako računalniško podpreti komuniciranje med posamezniki (Winograd 1986).

## 3 ALI LAHKO RAČUNALNIK ODLOČA IN MISLI ?

Kljub vsemu pa ne smemo pozabiti, da je računalnik le stroj in zato od njega vseeno ne smemo pričakovati preveč (Kajzer, 1987). Kakor vemo, je računalnik še daleč od vsemogočnega "mislečega" stroja, ki bi lahko zamenjal človeka, vendar pa ga nikakor tudi ne smemo podcenjevati: na številnih področjih človeka odločilno prekaša in je (lahko) nenadomestljivo orodje. Kako smotrno ga bomo znali uporabiti, pa je odvisno od nas samih. Nevarno je, na žalost zelo razširjeno mnenje, da lahko vse zmogljivejša računalniška tehnika in tehnologija zamenja človeka ne le pri rutinskem, temveč tudi pri ustvarjalnem delu. Kakor ugotavlja Kajzer (Kajzer, 1987): "Če le nekoliko pretiravamo, bi to pomenilo, da bodo vse bolj "butasti" ljudje z vse "pametnejšimi" računalniki čedalje ustvarjalnejši. Kakšna zabloda! V resnici je položaj ravno obraten: čim boljše in učinkovitejše je orodje, tem več mora biti v glavi in roki, ki ga uporabljata."

Po začetnih naporih "umetne inteligence" in računalniku kot "mislečem" stroju (Moto-Oka, 1984), se je tovrstni zanos nekoliko umiril, kajti vse bolj je jasno, da je računalnik le stroj in bo to vedno tudi ostal. Odgovoru na to vprašanje se je mogoče približati, če razmislimo, kaj pravzaprav pomeni pojem inteligenca. Ta uganka vznemirja filozofe ža od nekdaj in resnično je ni lahko razvozlati. Jasno je, da lahko inteligenco obravnavamo z več vidikov, npr. kot sposobnost načrtovanja, učenja, reševanja problemov, uporabe in razumevanja jezikov itd. Obnašanje, ki kaže (vsaj) eno izmed teh sposobnosti, šteje večina ljudi za inteligentno. Cilj tistih, ki se ukvarjajo z umetno inteligenco, je torej pripraviti računalnike do tega, da se bodo sposobni tako obnašati. Zato so tudi izhodišča ekspertnih sistemov, kot "nadomestka" umetne inteligence postavljena nekoliko drugače (Schnupp, 1985). Izhajajo iz predpostavke, da lahko razvijemo sisteme človek-računalnik, v katerih bi se računalnik obnašal "inteligentno", vendar ne na takšen način kot to počne človek.

Menimo, da lahko računalnik odloča le v primerih, ko zadošča obnavljanje vnaprej pripravljenega postopka. V teh primerih lahko govorimo o programiranem odločanju (Simon, 1960). Odločanje lahko programiramo le takrat, kadar lahko opredelimo natančna pravila, po katerih se bo odločanje odvijalo. Ta pravila so osnova za računalniški program, ki bo "prevzel" odločanje in s tem nadomestil človekovo presojo. Računalnik pa v tem primeru ne bo mislil, bo le obnavljal, kar pomeni v besednjaku kognitivne psihologije učenje in ne mišljenje.

Zaključimo naše razmišljanje s tem, da gre pri vprašanjih, ali lahko računalnik odloča in ali lahko računalnik misli, za dve različni stvari. Računalnik lahko odloča, vendar ne na način, ki ga označujemo kot miselni proces - mišljenje.

## 4 MODELI RAČUNALNIŠKE PODPORE POSLOVNEGA ODLOČANJA

Računalniško podporo poslovnega odločanja je vedno bila in bo tudi v bodoče predvsem heterogena celota, nikakor pa ne homogena oziroma monolitna "tvorba", čeprav bi to mnogi želeli. Zaradi tega jo moramo na različne načine organizirati. Predvsem pa moramo spoznati, da računalniška podpora odločanja nikakor ni homogena celota, kot smo še donedavnega mislili. Menimo, da jo moramo, v kolikor hočemo ponazoriti omenjeno heterogenost, ter ugotoviti, kje smo in kam gremo, analizirati vsaj z vidikov:

(1) metod odločanja, ki jih želimo računalniško podpreti (analitične metode <---> hevristične metode);

(2) stopnje presoje, ki jo prepustimo človeku (malo človekove presoje <---> veliko človekove presoje);

(3) števila posameznikov, ki sodelujejo pri (računalniško podprtem) poslovnem odločanju (individualno poslovno odločanje <---> skupinsko poslovno odločanje).

Ob upoštevanju omenjenih treh vidikov, lahko torej ločimo naslednjih pet tipov računalniške podpore poslovnega odločanja, ki se med seboj, čeprav služijo istemu namenu, v mnogočem razlikujejo:

(1) Računalniška podpora tip I podpira analitične metode, ne dopušča veliko človekove presoje in je namenjena podpori individualnega poslovnega odločanja. Gre za računalniško podporo odločanja, ki jo pogosto označimo kot strukturirano poslovno odločanje (angl. SDS) - odločevalec izbere model, po katerem bo računalnik odločal (Gorry, 1976).

(2) Računalniška podpora tip II podpira analitične metode, dopušča veliko človekove presoje in je namenjena podpori indivdualnega poslovnega odločanja. Tovrstno računalniško podporo poslovnega odločanja najpogosteje imenujemo sistem za podporo odločanja (angl. DSS). Opredelimo jo lahko kot tisto, ki odločevalca soočenega s semistrukturirano odločitvijo, podpira skozi dialog človek-računalnik, možnostjo učinkovite uporabe podatkov in vrste modelov za analizo

problemov ter sintezo odločitev (Sprague 1983).

(3) Računalniška podpora tip III podpira hevristične metode, ne dopušča veliko človekove presoje in je namenjena podpori individualnega poslovnega odločanja. Danes najbolj razširjen predstavnik tovrstne računalniške podpore so ekspertni sistemi (angl. ES), ki odločevalcu sugerirajo odločitev, oblikovano na podlagi v računalniku strukturiranega (strokovnega) znanja in posplošenih, problemsko neodvisnih hevrističnih metod (Krallman 1987).

(4) Računalniška podpora tip IV podpira hevristične metode, dopušča veliko človekove presoje in je namenjena podpori individulnega poslovnega odločanja. Gre za ekspertne sisteme za podporo odločanja (angl. ESS), ki za razliko od ES dopuščajo izbiro hevristične metode (Luconi 1986).

(5) Računalniška podpora tip V je namenjena podpori medsebojno (so)odvisnega poslovnega odločanja posameznikov, ki sodelujejo, najpogosteje skozi semistrukturirano poslovno odločanje kot skupina (angl. GDSS) (DeSanctis, 1985).

Večina omenjenih modelov računalniške podpore poslovnega odločanja je sicer raziskanih, z ostalimi pa se informatika kot znanost intenzivno ukvarja. Menimo pa, da je kar (pre)pogosto kontekst njihove uporabe bližji upravljalnemu informacijskem sistemu (UIS), kot pa omenjenemu sistemu podpore odločanja SPO (glej točko 1). Zato tudi menimo, da jih moramo (v nekaterh primerih) postaviti na prava izhodišča, predvsem pa poiskati njihove stične točke in tako ugotoviti, kako jih povezati v heterogeno celoto, ki bo omogočala uspešno in učinkovito celovito podporo poslovnega odločanja.

Računalniška podpora tip I je "klasična", najstarejša, zato je na tem področju tudi relativno največ narejenega. Zakaj relativno ? Zaradi tega, ker je sicer resnično veliko narejenega, vendar je ravno tukaj kontekst uporabe računalnika največkrat sporen. Relativno nekaj manj je narejenega pri računalniški podpori tip II. Zakaj prav tako relativno ? Veliko vprašanj je namreč še odprtih, čeprav je koncept uporabe računalnika najpogosteje pravi. Še manj je narejenega na področju računalniške podpore tip III. Na ostalih področjih (tip IV in V) pa je narejenega resnično zelo malo.

Vemo, da se tako sistemi za podporo odločanja (tip II), kot podobno tudi ekspertni sistemi (tip III), nekako razvijajo v ekspertne sisteme za podporo odločanja (tip IV). Prav tako pa se predvsem sistemi za podporo odločanja (tip II) usmerjajo na področje računalniške podpore skupinskega odločanja (tip V).

## 5 IN KAKO NAPREJ ?

Ker je poslovni sistem množica na določen način med seboj povezanih posameznikov, oziroma množica skupin posameznikov, ki sodelujejo z namenom doseganja postavljenih (skupnih) ciljev, je model informacijskega sistema, ki smo ga označili kot sistem podpore odločanja (SPO), z vidika računalniške podpore dejansko množica medseboj povezanih sistemov človek-računalnik. V tem primeru torej lahko (in moramo) govoriti o nekakšni mreži sistemov človek-računalnik.

Vspostavitev sistema podpore odločanja (SPO), kot mreže sistemov človek-računalnik naj upošteva tako kategorijo odnosov človek-človek, kot tudi kategorijo odnosov človek-računalnik ter kategorijo odnosov računalnik - računalnik. Vse odnose moramo opazovati v kontekstu organizacije. Menimo, da so predvsem prvo, pa tudi drugo omenjeni odnosi predvsem (nikakor pa ne izključno) problem (ekonomske) informatike, zadnje omenjeni kot tudi drugo omenjeni odnosi pa predvsem (vendar tudi nikakor ne izključno) problem računalništva. Vsekakor smo postavljeni pred velik izziv.

LITERATURA:

/1/ Chorafas D.: Fourth and Fifth Generation Programming languages, McGraw-Hill, New York, 1986.

/2/ DeSanctis G., Gallupe: Group DSS: A New frontier, Data Base, 1985 Winter, str. 3-1Ø.

/3/ DeSanctis G., Gallupe: Information systems support for group decision making, MISRC-WP-85-1Ø, Univ. Minessota, 1984.

/4/ Gorry,G.A., Scott-Morton M.S: A Framework for MIS, Sloan Management Review, 1976/1, str. 55-7Ø.

/5/ Kajzer S., Marn F.: Cesa računalnik v poslovnem sistemu ne zmore, Organiziranje in razvijanje informacijskih sistemov, Radenci, Društvo ekonomistov Maribor, Maribor, 1986.

/6/ Kajzer S. in sodelavci: Raziskava izhodišč za oblikovanje modela poslovnih informacijskih sistemov, VEKS Maribor 1986.

/7/ Keen P.G.W: Interactive computer systems for managers: A modest proposal, Sloan Management Review, 1976/3.

/8/ Krallman H.: Betriebliche Entscheindungs-Unterstutungsysteme. Zeitschrift Fuhrung + Organisation, 1987/2, str. 1Ø9-117.

/9/ Luconi F.L., Malone T.W., Scott-Morton M.S.: Expert Systems:The Next Chalenge for mangers, Sloan Management Review, 1987/4, str. 3-15.

/1Ø/ Moto-Oka T., Kitsuregawa M.: The Fifth Generation Computer, Wiley, New York, 1984.

/11/ Schnupp P. in Leinbrandt U.: Expertensysteme, Nicht nur fur Informatiker, Springer, Berlin, 1985.

/12/ Simon H.A.: The New Science of Management Decisions, Harper & Row, New York, 196Ø.

/13/ Sprague R.H., Carlson : Building effective DSS, Prentice Hall, Englewood Cliffs, 1983.

/14/ Winograd T., Flores: Understanding Computers and Cognition, Ablex Publ. Corp., Norwood, New Jersey, 1986.

UDK 519.876 UUCP

Boris Domajnko
Iskra Delta

uucp network software is a part of UNIX distribution. uucp ( Unix to Unix CoPy ) is a system, compound of files and programs, used for network file transfer, mailing, remote command execution and direct access to adjacent computer. Conformation with OSI reference model is important for each computer network - it assures among other qualities a possibility of further improvements in the future. Otherwise its usage will cease and other protocols with internationally accepted standards will be implemented instead. A short analysis of uucp is presented in this paper with an aye on the OSI model though there was no ISO/OSI model known in the time when uucp was created. But uucp communication tools will still be used for some time because of existing networks. Three of them are mentioned: american UUCP, european EUnet and Yugoslav ZUUTA.

Del standardne distribucije UNIXa je uucp mreža. uucp ( Unix to Unix CoPy ) je iz datotek in programov sestavljen sistem, ki omogoča uporabo ukazov za prenos datotek in elektronske pošte med računalniki, izvajanje ukazov na oddaljenem računalniku ter direktno priključitev na sosednji računalnik. Ustrezanje OSI referenčnemu modelu daje vsaki računalniški mreži poleg ostalih kvalitet tudi možnosti za nadaljni razvoj. V nasprotnem primeru jo bo slej ko prej potrebno zamenjati s sodobnejšim konceptom. V času nastanka uucp-ja ISO/OSI referenčni model še ni bil znan. Kljub novim rešitvam pa bo še nekaj časa možna njegova uporaba zaradi že obstoječih mrež. Omenjene so tri: UUCP, EUnet in ZUUTA .

## 1. Uvod

Unix je operacijski sistem, ki se še razvija v obliki novih verzij in zahtevnejših standardov, npr. AT&T System V in POSIX - IEEE standard v fazi razvoja. Mreže so bile dolgo časa največji problem tega operacijskega sistema, saj je UNIX šele s kasnejšimi verzijami postal močno okolje za razvoj in izvajanje mrežnih aplikacij. Še iz prvih časov obstaja uucp mreža, ki je del standardne distribucije UNIXa. Oglejmo si jo zaradi njene preprostosti, razširjenosti in možnosti takojšnje uporabe.

## 2. uucp sistem

uucp ( Unix to Unix CoPy ) je sistem, sestavljen iz datotek in programov, ki omogoča prenos datotek in elektronske pošte med dvema računalnikoma, izvajanje ukazov na oddaljenem računalniku ter direktno priključitev na sosednji računalnik. Spada v skupino mrež, ki so globalne glede na določen operacijski sistem (2).

Minimalna konfiguracija za uucp mrežo je sestavljena iz dveh računalnikov, ki ju povezuje serijski kanal V.24 ( amer. RS-232C ).

Za vzpostavitev zveze sta možna dva načina. Pri prvem je naš računalnik gostitelj in omogoča, da se nanj priključi oddaljeni računalnik. Drugi način se izvaja takrat, ko naš računalnik kliče oddaljeni računalnik ( npr. preko javnega telefonskega ali paketnega omrežja ).

Kadar za povezavo uporabljamo asinhroni serijski kanal, lahko izvajamo samo en način naenkrat. Če uporabljamo samo en serijski kanal, je običajno na voljo za klic iz oddaljenega računalnika. V primeru, ko hočemo klicati ven, začasno prepovemo dospele klice in si rezerviramo kanal. Slednje velja le, če uporabljamo modem.

## 3. Priprava uucp sistema in opis delovanja

Ime uucp je bilo v začetku povezano s transportnim nivojem pri povezavi UNIX oz. sorodnih računalnikov. Mreža deluje tako, da sosed predaja sporočilo sosedu, dokler sporočilo ne pride do končnega naslovnika. uucp je ena od redkih mrež, kjer izvor določi pot podatkov. Pri večini ostalih mrež je usmerjanje dinamično in ne fiksno.

Kot že rečeno, je uucp sistem datotek in programov. Datoteke vsebujejo informacije o mreži oz. o vlogi našega računalnika v njej, prav tako pa tudi informacije o podatkih, ki jih hočemo prenašati. Datoteke moramo imeti ločeno za oba načina - klicni in klicani. Oglejmo si nekaj bistvenih datotek, da osvetlimo informacije, ki jih ima računalnik v zvezi z uucp mrežo.

Datoteka **/etc/systemid** hrani ime lokacije ( kraj ) in ime našega računalnika.

V datoteki **/etc/passwd** so opisani vsi uporabniki računalnika in njihova kodirana gesla. Poleg tega je tu še ime programa, ki naj se začne izvajati, ko uporabnik pove svoje ime ( login ). Za klice iz mreže torej tu zaznamujemo avtomatičen start programa, ki bo bedel nad njimi.

Datoteka **/usr/lib/uucp/USERFILE** vsebuje omejitve v zvezi z zunanjim dostopom do našega računalnika, npr. do katerih direktorijev ima določen uporabnik iz mreže dostop.

Za pomoč pri dostopih do drugih računalnikov pa nam služi datoteka **/usr/lib/uucp/L.sys**, ki vsebuje podatke o računalnikih, ki jih lahko kličemo, npr.: ime računalnika, dneve z dovoljenim dnevnim časom dostopa, vrsta in številka kanala, hitrost, telefonsko številko ter določene informacije za postopek prijavljanja na oddaljeni računalnik.

Za jasnost podatkov v prej omenjeni datoteki imamo še datoteko **/usr/lib/uucp/L-dialcodes** , ki vsebuje okrajšave za določene telefonske številke, ki jih pogosto uporabljamo.

Poglejmo še najvažnejše programe v sistemu uucp.

Če hočemo prenašati datoteke, jih prepustimo programu **uucico** ( Unix to Unix Copy In Copy Out ). Le-ta izvrši delovne datoteke za prenos podatkov, ki jih je pripravil uucp. Ima pa tudi glavno vlogo pri prenosu v obratno smer, torej pri sprejemu. Klic običajno izvajamo periodično in avtomatično, tako da definiramo v datoteki **/usr/lib/crontab**, kdaj naj poseben sistemski program kliče uucico. Ta ob izvajanju pogleda, če ima na določenem direktoriju zahteve za prenos. Če obstajajo, izvrši prenose. Prenos se lahko izvede tudi v obratni smeri, če na oddaljenem računalniku čakajo zahteve za prenos na našega. Uporabniku tako ni treba skrbeti za vzpostavitev zveze, zadošča mu, da ve, da se prenos izvaja npr. vsake pol ure ali pa npr. v nočnih urah.

Program **uucp** poskrbi za prenos datotek, tako da najprej kreira določene delovne in podatkovne datoteke. Če prenos ni možen takoj, ga bo uucico poskušal izvesti v časovnih intervalih, ki so definirani v datoteki **crontab**.

**uux** izvrši ukaz na določenem računalniku. Datoteke, ki so povezane z ukazom, lahko prečita na različnih računalnikih. Poglejmo za primer, kako primerjamo dve datoteki z imenom dat. Prva se nahaja na računalniku t1, druga na računalniku t2, rezultat pa naj bo shranjen na našem računalniku v datoteki rez. Običajna ukazna vrstica bi bila taka-le:

```
diff /usr/a/dat /usr/b/dat > rez
```

Ukazna vrstica z uux ukazom je podobna običajni, le da dodamo ukaz uux ter definiramo imena računalnikov, kjer se posamezne datoteke nahajajo:

```
uux "diff t1!/usr/a/dat t2!/usr/b/dat > !rez"
```

**uuxqt** izvrši ukaz, ki pride iz oddaljenega računalnika. Preveri, če je ukaz dovoljen, saj posameznim zunanjim uporabnikom lahko zelo omejimo pravice glede klicanja programov na našem računalniku.

**mail** je sistem za elektronsko pošto. Obsega pripravo sporočil in pošiljanje oz. sprejemanje sporočil. Uporablja uucp sistem.

**cu** je program, ki nam omogoča, da se kot terminal priključimo na sosednji računalnik.

4. uucp in OSI referenčni model

Protokoli v računalniških mrežah so lahko zelo kompleksni. Da jih lažje obvladamo, so izdelani v različnih nivojih. Spodnji so bolj povezani s strojno opremo, najvišji pa predstavljajo stik z uporabnikom. Nivoji so definirani ob takšnih zahtevah (7):

- nov nivo je kreiran tam, kjer je potreben nov nivo abstrakcije,

- vsak nivo mora izvajati dobro definirane funkcije,

- funkcije vsakega nivoja morajo biti izbrane v smislu definiranja oz. mednarodno definiranih protokolov,

- meje nivojev morajo biti izbrane tako, da je pretok informacij med njimi minimalen,

- število nivojev mora biti dovolj veliko, da ni treba mešati določenih funkcij, in dovolj malo, da celoten sistem ne postane preveč zapleten.

UNIXov uucp protokol je bil izdelan za povezave, ki niso stalne. V času njegove definicije OSI referenčni model še ni bil znan in če smo zdaj za implementacijo ISO standardov, moramo sprejeti uporabo novih protokolov (8). V začetni fazi pa je še vedno potrebna možnost uporabe standardnih UNIXovih komunikacijskih orodij ( cu, uucp ) in aplikacij ( npr. mail ) preko asinhrone ali X.25 linije. To je nujno potrebno že zaradi nemotenega delovanja obstoječih mrež, dokler se le-te ne prilagodijo standardom.

Programska oprema za uucp je sestavljena iz več delov. En del služi za pripravo zahtev za prenos, drug del nastavi karakteristike kanala in vzpostavi zvezo ob uporabi pravilnega gesla, na koncu pa tudi zaključi zvezo. Tretji del pa se izvaja med prenosom in pozna dva načina: klicni in klicani. Poskrbi za oddajo in sprejem datotek, prav tako zna sprejete podatke poslati naprej proti končnemu naslovniku. Uporabljen je lasten protokol s kontrolo pravilnosti podatkov. Če je spodnji nivo zanesljiv ( npr. X.25 ), to kontrolo izpustimo.

Naredimo zdaj poskus podreditve funkcij uucp-ja ISO/OSI modelu, kar ni lahko, saj le s težavo ločimo funkcije 5., 4., in 3. nivoja.

7. APLIKACIJSKI NIVO

Pravzaprav lahko celoten uucp sistem imenujemo aplikacijski program, saj omogoča ( tudi drugim programom ) prenos datotek in elektronske pošte ter priključitev na sosednji računalnik.

6. PREDSTAVITVENI NIVO

Ker uucp ni mišljen za sodelovanje heterogenih sistemov, ne omogoča prilagajanja lokalne sintakse, razen pri prenašanju neznakovnih datotek. Ostali programi za obdelavo datotek pred oddajo oz. po sprejetju niso del uucp sistema in jih izvajamo posebej ( npr. kriptografska obdelava ). Pri oddaji pa vseeno lahko ustrezne ukaze definiramo na istem mestu kot ostale ukaze za pošiljanje.

5. NIVO SEJE

Nivo seje nudi funkcije za povezavo dveh aplikacijskih procesov v odnos logične komunikacije in funkcije za organizacijo in sinhronizacijo njunega dialoga. Preden dva aplikacijska procesa začneta komunikacijo, morata vzpostaviti komunikacijo na nivoju seje. Pri uucp-ju je to pogovor dveh programov uucico, ki se običajno ne izvaja takoj ob zahtevi za prenos , ampak ob določenih časovnih intervalih. Tako lahko pozabimo na dvosmerno sočasno razpravo med dvema aplikacijskima procesoma. Pomembno vlogo imajo informacije, ki so shranjene v različnih datotekah:

- dovoljenja za dostope iz drugih računalnikov,
- dovoljenja za izvrševanje ukazov iz mreže in seznam dovoljenih ukazov ( za vsakega zunanjega uporabnika posebej ),
- datoteke s podatki o povezavah z drugimi računalniki in datoteka okrajšav imen.

Povezava bo razpuščena, če ne dobimo odgovora v določenem času. Ves potek dialoga se zapisuje v dnevnik, t.j. v posebno datoteko.

---

## 4. TRANSPORTNI NIVO

Pri uucp-ju ni prave meje med transportnim in mrežnimi nivoji, detajli transportnega protokola pa so poleg vsega še nedokumentirani. Ta nivo je na voljo tudi za pošiljanje elektronske pošte ( "mail"), ki je vezano na uucp sistem. Na splošno za poštni del velja, da je narejen eksplicitno z aplikacijskimi protokoli, implementiranimi z uporabo izvrševanja ukazov na oddaljenem računalniku.

Transportni nivo po pravilih dobi podatke iz nivoja séje in jih oblikuje v pakete. Ker gre pri uucp-ju pravzaprav za isti program, nivoja v zvezi s čistimi podatki težko ločimo. Ko preda podatke mrežnemu nivoju ( krmilnemu programu za serijske linije ), čaka potrditev od prejemnika. Vključena je tudi kontrola prenosa. V primeru uporabe X.25 protokola kontrole pravilnosti podatkov na tem nivoju ni zaradi hitrejšega prenosa. Transportni nivo odigra svojo funkcijo tudi v primeru, ko ugotovi, da ni končni naslovnik in mora zato prejete podatke poslati naprej.

Spodnji trije nivoji so odvisni od metode prenosa podatkov v mreži ( navadna serijska linija, V.21 modemska povezava, X.25, LAN).

---

## 3. MREŽNI NIVO

Identiteta končnega naslovnika ni zadostna informacija, saj ne uporabljamo običajneg dinamičnega usmerjanja, ampak fiksno. Izvor sam določa pot podatkov, transportni nivoji na njihovi poti pa vseeno imajo možnost, da po svoji lastni presoji to pot spreminjajo. Takšno fiksno usmerjanje potrebuje bazo s podatki za vse povezave. Računalnik sam iz te baze ugotovi najprimernejšo pot, tako da je uporabniku ni treba vnašati ročno. Problem je v zamudnem sestavljanju baze, ki se izvaja na enem računalniku za celotno mrežo in mora biti čim bolj pogosto, saj moramo slediti spremembam v konfiguraciji celotne mreže.

---

Tak bi bil opis standardnega uucp-ja. Na preprostem modelu lahko mesto prvotnega uucp-ja prikažemo tako:

| |
|---|
| elektronska pošta, novice, prenos datotek |
| uucp |
| V.21 |

Tak uucp, kakršen je prikazan na gornjem modelu, z OSI modelom pravzaprav nima zveze. Člani projekta ROSE (8), ki predstavlja integracijo določenih proizvajalcev s ciljem, da združijo moči pri prilagoditvi UNIXa OSI nivojem in s tem omogočijo priključitev na drugačne računalnike in mreže, če le ustrezajo mednarodnim standardom, so predvideli uucp samo še v vmesni fazi svojega projekta. Z uporabo uucp-ja preko X.25 in LAN mreže so se izognili klasičnemu problemu pri implementaciji OSI protokolov, t.j. pomanjkanju ustreznih servisov v obdobju, ko še ni implementiran protokol na vseh nivojih.

| | |
|---|---|
| elektronska pošta, novice, prenos datotek | |
| uucp | |
| ISO transportni razred 2/3 | ISO transportni razred 4 |
| X.25 | LAN CSMA/CD |

V naslednji fazi pa se uucp izgubi med funkcijami, ki jih natančno določa ISO standard. Temeljito izdelan ISO nivo seje in posebna knjižnica potrebnih funkcij omogočata kasnejšo izdelavo drugih aplikacij, ne samo obeh spodaj omenjenih. Mreže posameznih družb, ki ne podpirajo OSI modela, se bodo priključile preko posebnih računalnikov ( Gateway ), posamezni PTT servisi pa do tistih OSI protokolov, ki jih podpirajo.

| | | | |
|---|---|---|---|
| 7,6 | dostop do datotek ali njihov prenos ( ISO FTAM ) elektronska pošta in konferenca ( X.4xx ) | | |
| 5 | ISO nivo seje ( ISO 8326/7 ) ( funkcije v jedru operacijskega sistema ) | | |
| 4 | ISO transportni razred Ø/2/3 | ISO transportni razred 4 | Gateway |
| 3,2,1 | X.25 | LAN CSMA/CD | |

## 5. UUCP mreža v ZDA

UUCP je ime ene od velikih nacionalnih mrež v ZDA. Mreža spada med "mreže uporabnikov s podobnimi interesi". Njeni prvi začetki segajo v leto 1978. Povezavo geografsko oddaljenih računalnikov v večini primerov še zdaj omogoča telefonsko omrežje. Prenos datotek in izvrševanje ukazov na oddaljenem računalniku sta bila osnovni namen take povezave, možnost pošiljanja elektronske pošte pa se je pojavila kasneje (3). To možnost so lahko začeli uporabljati tudi računalniki z drugačnimi operacijskimi sistemi. Prilagajanje sintakse imen zahtevam ARPA Internet mreže kaže na interes za boljšo povezaveoz ostalimi mrežami, daje pa tudi možnosti za izvedbo boljšega usmerjanja.

Centralna administracija v UUCP mreži ne obstaja. Stanje bi lahko primerjali z anarhijo. Bolje so organizirani manjši deli mreže, ki jih povezuje večji medsebojni promet. Za priključitev na mrežo je sicer dovolj, da najdemo soseda, ki je že v mreži in ki se strinja, da bo naš sosed. Svoj naslov še vnesemo v poseben seznam, da nas bodo lahko našli tudi drugi računalniki. Če imamo svojo lokalno mrežo, dodelimo računalnikom znotraj nje večje pravice, enega pa določimo za povezavo z ostalim delom UUCP mreže. Tako nam ni treba podvajati podatkov o zunanji mreži, ker pa imajo računalniki iz zunanje mreže dostop v našo le preko enega, na enem mestu lažje skrbimo za izvajanje varnostnih aspektov.

Vsak računalnik plačuje za svoje povezave. Zato nekateri podpirajo priključitev čim večih sosedov, saj si s tem zagotovijo krajšo pot za lastna sporočila. Stroški mreže so v glavnem povezani s plačilom telefonskih storitev.

Če upoštevamo še sorodne mreže, s katerimi je povezan, je v UUCP mrežah medsebojno povezanih med 7000 do 10000 računalnikov (3).

## 6. Evropska UNIX mreža - EUnet

Iz UUCP mreže se je leta 1980 razvila v ZDA še USENET, ki omogoča med drugim tudi X.25 povezave. Leta 1982 pa je združenje EUUG ( European Unix systems User's Group ) ustanovilo evropsko mrežo EUnet ( "European UNIX net" ). Ker utegne biti zanimiva tudi nas, si jo oglejmo.

EUnet ima za vzor UUCP in USENET iz Severne Amerike. Večina računalnikov ima UNIX operacijski sistem. Organizacija mreže je boljša kot v Severni Ameriki, saj za njo stoji organizacija EUUG. Mreža je neposredno ali posredno priključena na ostale evropske in neevropske mreže. Njen glavni namen je pošiljanje elektronske pošte in novic (4), preko X.25 pa je možna tudi direktna priključitev na oddaljni računalnik.

V vsaki državi je po en glavni računalnik, ki skrbi za mrežo v svoji državi, včasih kar z direktno povezavo z vsemi ostalimi računalniki. Glavni računalniki posameznih držav lahko komunicirajo med seboj. Obstaja še glavni računalnik za celo mrežo, ki povezuje glavne računalnike posameznih držav - to je računalnik **mcvax** v Amsterdamu. Ta računalnik je povezan z ostalimi UNIX mrežami : ACSnet ( Avstralija ), USENET ( ZDA ), CDAnet ( Kanada ), JUNET ( Japonska ), SDN ( Koreja ) in še mreži v Izraelu in Novi Zelandiji. mcvax pa ima povezave še z mnogimi drugimi mrežami, ki ne bazirajo na UNIX operacijskem sistemu. Prehod na druge mreže lahko nudijo tudi nekateri drugi računalniki v mreži.

Ta mreža se širi. Najprej so jo uporabljali entuziasti, ki so si pošiljali sporočila in imeli na ta način skupne debate. Sedaj je krog uporabnikov dosti večji. To so univerze, posamezna podjetja ter reklamna podjetja. Povezuje jih tudi EUUG združenje, ki svojim članom ( uporabnikom UNIXa ) tudi omogoča dostop do različne programske opreme. Izdajajo svojo revijo, ki je vsa namenjena UNIXu. V okviru združenja tečejo tudi razgovori o prilagoditvi ISO/OSI modelu na nivoju komunikacije med meddržavnimi vozlišči.

V Evropi je priključenih približno 800 računalnikov iz 16. držav.

## 7. Sklepne misli

Videli smo, da uucp zagotavlja možnost povezave v mrežo za računalnike UNIXove družine. Kot tak bo počasi izgubil svoj pomen, saj ne ustreza ISO/OSI referenčnemu modelu. V teku so projekti, ki imajo za cilj ne samo prilagoditi UNIX mreže ISO in CCITT standardom, temveč na ta način tudi zagotoviti kvalitetno povezavo z računalniki, ki imajo drugačne operacijske sisteme. Tak razvojni projekt je npr. ROSE (8), ki ga podpira ESPRIT Information Exchange System (IES). Trenutno se ukvarjajo s prototipom OSI okolja pod UNIX System V in Berkeley 4.2BSD. Sicer pa zaenkrat pripada večja pozornost skupini protokolov, ki v splošnem ustreza 3. in 4. nivoju ISO/OSI referenčnega modela, čeprav je bil slednji narejen pozneje. To je TCP/IP ( Transmission Control Protocol / Internetwork Protocol ). Primeren je tudi za povezavo računalnikov z različnimi operacijskimi sistemi.

Pravzaprav UNIX šele z novimi verzijami omogoča izvedbe kompleksnih mrež s skupno uporabo datotek in periferne strojne opreme. Funkcije in mehanizmi, ki omogočajo enotno vpeljavo različnih protokolov in podporo različnim medijem prenosa in ki so prej manjkali v operacijskem sistemu, so zdaj končno na voljo. Novosti "streams" in "sockets" (5) sta dodaten mehanizem poleg že obstoječih krmilnih programov ( "device drivers" ). AT&T je npr. razvil poenoten vmesnik transportnega nivoja ( Transport Level Interface ), ki nam omogoča, da lahko pišemo aplikacije, ki so neodvisne od mreže. Zato se tudi nova verzija uucp-ja lahko izvaja neodvisno od mreže, ki se nahaja v spodnjih nivojih. S tem pa je uucp postal le še ena od možnosti uporabe mreže na najvišjem, to je aplikacijskem nivoju. Družbo mu delata npr. še dva protokola, ki sta od spodnjih nivojev neodvisna in ki omogočata preprost in hiter dostop oziroma souporabo datotek - RFS ( Remote File Sharing, AT&T ) in NFS ( Network File System, Sun Microsystems ).

Povrnimo se še k mreži EUnet, saj se širi tudi k nam. Letos osnovano Združenje uporabnikov UNIX tehničnih aplikacij ( ZUUTA ) pripravlja javno uucp mrežo, ki bo zaživela jeseni. Uporabniki mreže si bodo poleg elektronske pošte lahko medsebojno izmenjevali najrazličnejše novice, informacije, izkušnje in programe. Možna bo povezava s svetom, na domačem terenu pa bodo imele možnost tudi delovne organizacije, ki so zainteresirane za napredek na tem področju, saj bodo imele možnost stalnega kontakta z uporabniki.

Viri:

1. XENIX Communications Guide, 1984 Intel
2. A. Weiskopf: Networking Products Break New Ground, Unix/World, I/1987
3. Quaterman & Hoskins: Notable Computer Networks, Communications of the ACM, Vol.29, Num. 10, 1986
4. Houdler: EUnet, EUUG Newsletter, Vol.7, Num.2, 1987
5. Coffield & Sheperd: Tutorial guide to Unix sockets for network communications, Networks, Vol. 10, No. 1, 1987
6. UNIX System V - Release 2.0 Administrator Guide, AT&T, 1984
7. Zimmerman, H.:OSI Reference Model, IEEE Trans. Commun. Vol. COM-28, April 1980
8. The ROSE Consortium: ROSE, EUNET and the Migration to OSI, EUUG Newsletter, Autumn '86

# INFORMACIJA IN INFORMACIJSKI SISTEM*

UDK 519.724

**Anton P. Železnikar**
**Iskra Delta**

V eseju se obravnavajo tale naslovna vprašanja: odpiranje nove (informacijske) konverzacije, filozofsko in biološko ozadje informacije (informacija kot oblika, vsebina, proces in pojav; ontološko razumevanje informacije; koncept informacije kot nastajajoče in razvijajoče pojavnosti), informacija kot princip in kot sistem v živem, umetni informacijski sistemi (artificialno kot folozofija, znanost in tehnologija; načrt in izvedba informacijskega sistema; informacijska povezava človek-okolje in človek-stroj), primeri živih in umetnih informacijskih sistemov (možnosti formalizacije in konstruktivnega obravnavanja informacijskih sistemov in pojavov).

INFORMATION AND INFORMATION SYSTEM. *This essay deals with the following topics: opening of a new (informational) conversation, philosophical and biological background of information (information as form, contents, process, and phenomenon; ontological understanding of information; a concept of information viewed through phenomenology of arising and developing), information as a principle and as a living system, artificial information systems (the artificial as philosophy, science, and technology; planning and designing of an information system; informational conectedness in the man-environment and man-machine interaction), some examples of living and artificial information systems (possibilities of formalization and constructive approaches in the domain of information systems and informational phenomena).*

## 0. Uvod

*Zival seveda ve. Vendar zanesljivo ne ve, da ve.*

Teilhard de Chardin

$\nvDash_K \alpha \vDash_K$; (glej pojasnilo na koncu spisa)

Anton P. Železnikar

Priročna knjižica [1], ki jo je izdalo podjetje IBM, ima naslov: *"Vse je informacija."* Fred I. Dretske [2] začenja svojo bolj filozofsko kot tehnološko in matematično študijo o znanju in informacijskem pretoku z domala gnostično premiso: *"V začetku je bila informacija. Beseda je prišla kasneje. Ta prehod se je izvršil z razvojem organizmov, z njihovo sposobnostjo izbiranja informacije za preživljanje in za nadaljevanje njihove vrste."* Hkrati pa ugotavlja, da *"... se nahaja semantika izven obzorja matematične teorije informacije"* in da *"... nam komunikacijska teorija ne pove, kaj je informacija. Ta teorija zanemarja vprašanja, ki zadevajo vsebino signalov in kakšno informacijo nosijo, da lahko opisuje, koliko informacije nosijo."* Morda je pomembno, da v okviru Dretskejeve študije omenimo tudi njegovo stališče o znanju: *"Kaj je znanje? Tradicionalni odgovor pravi, da je znanje oblika zagovarjanega prepričanja. ... Prepričanja so lahko lažna in resnici se lahko ne verjame. ... Ne pomaga tudi izgovor, kot se to večkrat dogaja, da je znanje odvisno od ustreznega zagovarjanja, če se pri tem ne pove, kaj je v svojem bistvu t.i. ustrezno zagovarjanje."*

Zakaj je potrebno vsa ta vprašanja navajati in kako jih je mogoče vmeščati v diskurz vpraševanja o informaciji in o informacijskem sistemu? Človeško bitje, ki velikokrat zagovarja svoja razmišljanja in vedenje s t.i. zdravim razumom, ostaja pogosto izven domene vpraševanja o informaciji. Ali se kdaj samokritično sprašujete nekako takole: *"Kaj je t.i. zdravi razum kot informacija razmišljajočega in v kulturno okolje prilagojenega bitja? Kako bitje v sebi kopiči, prečiščuje, izbira in preoblikuje informacijo kot informacijsko bitje, kaj z njo počenja in kako pri tem uporablja drugo svojo informacijo?"* Tu nastane vprašanje, s katero drugo informacijo in s katerim bitjevskim

* Ta spis je bil predstavljen na Posvetovanju o informacijskih sistemih na Tehniški fakulteti v Mariboru, dne 2.6. 1988 in objavljen v zborniku posvetovanja.

informacijskim sistemom. Torej, ali ima bitje svojo informacijo in informacijo, s katero to svojo informacijo oblikuje, jo sprejema in sporoča? Ce se o tem pogovarjamo s svojim zdravim razumom, nam ta odgovarja, da vsekakor imamo svojo informacijo in svoj sistem informacijske obravnave. Do naslednjega, bistvenega spoznanja je lahko le kratek korak: v bitju informacija nastaja kot življenska procesnost, kot regularna in neregularna (nova, mutacijska) pojavnost živega organizma. Tako sproti in nenehno oblikujemo informacijo ne le za tiste funkcije, ki ohranjajo življenje, temveč tudi za drugo svojo aktivnost, za vedenje, razmišljanje, sporočanje in informacijsko sprejemanje.

Kot informatiki, kot akterji in agenti na področju računalniških in informacijskih dejavnosti, kot socialna bitja informacijske epohe pa se nenehno soočamo s vpraševanjem, kaj je informacija in kakšen je njen sistem. Poudarjam, kakšen je njen sistem! Kakšne principe informacije je mogoče konstruirati, če ti ne čakajo, da bi lahko bili odkriti, temveč so kot principi, kot principska informacija lahko le informacijsko konstruirani? Ce tega vpraševanja ne izvajamo, smo dejansko še predljudje, živimo bolj nezavedno in naključno vrženo v okolje in v sami sebe in ne spoznavamo dovolj inteligentno t.i. lastne in socialne, medsubjektne informacijske pojavnosti. Informacija prav gotovo ni nekaj enostavnega, statičnega, neprepletenega, necirkulirajočega, je v stalnem nastajanju, kot da že tudi sama sebe oblikuje kot sistem. Je kot gost in gostitelj, kot objekt in subjekt v živih, nastajajočih in nenehno se spreminjajočih informacijskih agregatih. Informacijska doba prinaša novo razumevanje informacije, ki je nova razvojna stopnja človeka kot visoko strukturiranega in organiziranega bitja. *"Eden od namenov tega eseja je odpiranje nove konverzacije, s katero bi bilo mogoče artikulirati moč človekovega informacijskega uma,"* je bilo rečeno v [3]. Poudarek te izjave je v tem, da se končno informaciji kot fenomenologiji živega prizna tista njena narava, ki je že tisočletja spoznavno zasidrana v zdravem razumu človeka in ki sega iz pradavnine, starogrške filozofije, gnosticizma, evropskega racionalizma do novoveške filozofije in se danes kot še nikoli doslej pojavlja kot nov, zavesten in smiseln imperativ preživetja.

## 1. Filozofsko in biološko ozadje informacije

### *1.0. Informacija kot oblika, vsebina, proces in pojav*

Za živo bitje ima informacija pomen predvsem kot pojavnost v njegovem organizmu. Ta pojavnost, ki jo imenujemo informacijska, ima svojo oblikovnost (genetičnost, strukturnost), vsebinskost (odnosnost, organiziranost) oziroma kratkomalo procesnost. To pa sploh ne pomeni, da ne obstaja tudi neživa informacija, ki jo lahko opažamo v naravi, v kozmičnih pojavih in v človekovih tehnoloških in tudi že v bioloških artefaktih. V področje umetne informacije uvrščamo tudi računalniško bazo podatkov, računalniški program in računalniško arhitekturo, ki omogočajo fleksibilno (programirano) podatkovno transformacijo. S tem smo povedali le to, da je informacija informacijsko aktivna in pasivna pojavnost vesoljnega (naravnega), življenskega (biološkega) in artificialnega (konstruktibilnega, umetnega).

Informacijska oblika in informacijski proces sta vselej tudi vsebinska (pomensko spremenljiva in odvisna od razumevanja, ki ga bitje trenutno premore). Informacijski pojav je kot oblikovna in procesna entiteta (avtonomnost, združenost, zaokroženost, vase-zaključenost) nosilec takšne ali drugačne vsebinskosti (interpretivnosti, kodiranosti). S pojavom oblike in procesa se odpira neizčrpen in neponovljiv videz informacije. Oblika in proces lahko nosita (povzročata, vznemirjata) pojavno raznovrstna sporočila in začenjata nove, nastajajoče informacijske pojave in informacijsko oblikovanje (kreiranje). Informacija pomeni individualnost tako po obliki in/ali procesu. Senzorska informacija bitja se na svojih poteh do korteksov individualno (svojsko, nepredvidljivo, metafizično) preoblikuje v odvisnosti od obstoječe, totalne bitjevske informacije. Pojav, ki kot opaženo nekaj sporoča, se različno informacijsko interpretira v organizmu bitja in sproža različne informacijske akcije (reakcije, odločitve, voljo za vedenje, zavest). Informacija prinaša tako vselej še potencialno, neopaženo, neraziskano komponento in tako učinkuje na informacijsko okolje (domeno, carstvo), v katero vstopa.

Sodobna tehnologija in spoznanja v raziskavah procesnosti žive celice prinašajo zavest o informacijskem kozmosu, ki bi bil v tehnološkem in živem potencialno možen kot svojski (informacijski) sistem (nevralni, genetski in masivno-paralelni računalniki). Živa celica ima kompleksen, individualen, avtonomen informacijski sistem, s svojo logiko (dednostjo, mutacijskostjo, fleksibilnostjo) in tako svojsko procesnost, ki je značilna za življenski cikel celice, za njeno preživljanje, obrambo (imunost), prehranjevanje (metabolizem), replikacijo (razmnoževanje) oziroma samoprodukcijo (avtopoiesis). Na primeru žive celice je mogoče spoznavati, kako so v biokemijsko informacijsko strukturo (molekularno inteligenco) vmeščeni oziroma vgnezdeni drugi informacijski agregati, in sicer od oblikovno-pomnilnih (dednostna struktura DNA, RNA) do biološko-procesnih. V računalniških informacijskih sistemih imamo prav tako značilne pojave vmeščanja ali vgnezdevanja ene informacije (podatkovne strukture) v drugo. Tako je npr. izdelani uporabniški program najprej vmeščen v okolje jezikovnega prevajalnika, prevajalnik v okolje operacijskega sistema in operacijski sistem v okolje računalniške arhitekture. Vsi ti elementi računalnika so razumevani kot specifična informacija, in sicer za trenutno uporabo, prevajanje, operativnost in strojnost. Pri podatkovnih strukturah kot definicijsko statičnih informacijskih agregatih je princip vgnezdovanja lahko rekurzivna lastnost podatkovne strukture.

### *1.1. Ontološko razumevanje informacije*

Ontološko ali bitjevsko razumevanje informacije je v svojem filozofskem ozadju lahko fenomenološko, eksistencialno in funkcionalistično. Razumevanje nastajanja informacije je pojmovni paralelizem k nastajanju živega in je neposredno s tem razumevanjem povezano. Ontološko razumevanje informacije je preprosto vprašanje: *"Kaj je informacija."* Iz tega začetnega vprašanja se lahko razvija proces vpraševanja, ki se razširja v kajstvo biti informacije oziroma v vpraševanje o informacijski biti.

Clovek zmore gledati zavestno v svojo ali tujo informacijo in jo zmore tudi zavestno opazovati, raziskovati, spoznavati in razumevati. Tu je določena informacija objekt človekovega mišljenja, samo mišljenje pa informacijski subjekt, ki zmore opazovati svoje informacijske objekte. V tej svoji informacijski funkciji lahko mišljenje nastajalno (generativno), spontano (nepredvidljivo), cirkularno (rekurentno) in paralelno (prostorsko in časovno hkratno) transformira (manipulira in preoblikuje) informacijske objekte, ki so oblike in procesi kognitivne informacije. Tako nastaja ali se poraja vzvratno prepletena informacijska interakcija med informacijskimi subjekti (operatorji) in informacijski objekti (operandi), ko se hkrati izmenjujejo njihove objektivne vloge s subjektivnimi in obratno. Mišljenje se očitno manifestira (aktivno in pasivno informira) kot operacija in operand mišljenja. V človekovih korteksih je tako mogoče zavestno (introspektivno) opazovati subjektivno-objektivno procesnost informacijskih pojavov. Prav od tu, iz tega primera je tudi mogoče spoznavati spontano, nepredvidljivo, generativno, cirkularno, paralelno, posledično in vsakršnje nastajanje informacije v živem.

Ontološko razumevanje informacije je tedaj utemeljeno s prepričanjem, da je informacija tudi pojavnost živega in da zaobseže vprašanje o biti informacije prav za prav vsa druga vprašanja, ki se nanašajo na bit (nem. das Sein, angl. Being), na njeno eksistenco in njeno nastajanje. V okviru informacijske spoznavne zavesti je vse pojavljajoče in nastajajoče informacijsko fenomenološko, pri čemer pa se lahko v okviru filozofije informacije postavlja tudi vprašanje o informaciji po sebi. Seveda pa se z informacijskega vidika informacija kaže tudi v vseh kulturnih oblikah, npr. kot filozofija, znanost, umetnost, zavest, hotenje, prepričanje, umovanje, čustvovanje, sklepanje itd. Z informacijsko ontološkega vidika obstajajo le značilne informacijske oblike in značilni informacijski procesi, med katere se uvrščajo tudi vse kulturne, ideološke in religiozne pojavne oblike. Bitje se ob vseh teh informacijskih oblikah (kulturnih, socialnih, okoliških, kozmičnih) lahko identificira le s svojo celostno informacijo, ki jo je vobče mogoče označiti kot bitjevsko metafiziko. Takšno, kibernetsko pojmovanje informacije obstaja v bistvu (nezavedno) že v začetkih grške filozofije, v novejši dobi pa se informacija kompleksno kibernetično, semiotsko in sistemsko pojmuje že vsaj od nastanka Wienerjeve kibernetike.

### *1.2. Koncept informacije kot nastajajoče in razvijajoče pojavnosti*

Pri pojmovanju informacije nam gre torej za razumevanje prav tistega, kar sproti spontano, nepredvidljivo in seveda tudi pričakovano nastaja ali je šele v razvoju v živem (organizmičnem, bitjevskem), vesoljnem (kozmičnem) in artificialnem (umetnem, konstruktibilnem) in se pojavlja kot informacijsko. Informacijsko v živem je sedanjost nastajajočega iz preteklega (zgodovinskega) in iz videnja prihodnjega (prihajajočega). Bitjevska metafizika razpolaga s svojim modelom sveta (modelno informacijo) in s svojimi mehanizmi za upoštevanje tega modela (idelogije) pri nastajanju bitjevske informacije. Tudi razmišljanje o informaciji je lahko le metafizično, torej bitjevsko informacijsko, seveda dinamično prepleteno in vzvratno vnaprejšnje. Razmišljanje je lahko tudi bolj statično, vpreženo v znanstveno disciplinarnost, v ideološko omejenost, tako da je npr. deduktivno, aksiomatično, linearno sklepajoče, konsistentno, neprotislovno in v okviru svoje omejenosti tudi produktivno. Informacijska doba prinaša prav izziv informacijskega, svobodno, pluralno nastajajočega.

Razumevanje informaciske nastajalnosti pa prinaša nove možnosti prav tam, kjer so se doslej pojavljale nepremostljive ovire: v znastvenem, raziskovalnem in tehnološkem. Inteligenca kot informacijska lastnost živega je zaenkrat pojavnostno le nejasen ideal (cilj), ki se mu tehnološko in znanstveno poskušata približevati skozi disciplinarnosti umetno-inteligenčnega, spoznavno-teorijskega, nevroznanstvenega, psihološkega in filozofskega. Cloveka vznemirjajo danes raziskave inteligenčnega, žive inteligence, ki je kognitivno in spontano nastajajoča in se pojavlja kot bitjevska vrženost (nem. die Geworfenkeit, angl. throwness) v življensko problematične (usodne, nepredvidljive) položaje. Seveda je v tem pomenu informacijskega, zlasti nastajalnega prej ko slej potrebno razumevati (predvidevati) tudi drugačne (nove) možnosti abstrakcije, simbolične formalizacije in njene pomenskosti, ki naj bi presegale trdna, znastveno statična in s tem varna in produktivna (?!) zatočišča racionalistične tradicije.

## 2. Informacija kot princip in kot sistem

### *2.0. Informacijski principi*

Filozofske principe informacije je mogoče opredeliti intuitivno in jih potem (glej [4] in v tem spisu odstavek 4.2) tudi simbolično formalizirati. Poskusimo takole:

*(1)* Informacija je *cirkularen* in *spontan* pojav informacije, pri čemer sta tudi njena cirkularnost in spontanost informacija. Dva informacijska pojava (informacije, informiranja) sta vobče (zaradi njune spontanosti) informacijsko *različna*. □

*(2)* Informacija *nastaja* z informiranjem, in sicer s *protiinformiranjem* in z *vmeščanjem*

nastale informacije v obstoječo informacijo. Tudi nastajanje informacije je informacija. □

*(3)* Nastala informacija je *protiinformacija* (nova, še ne vmeščena informacija). □

*(4)* Informiranje informacije je prihajanje informacije v obstajanje (v eksistenco), njeno spreminjanje in izginjanje; to pa je prav *fenomenologija* informacije. □

*(5)* Z *vmeščanjem* nastale informacije oziroma protiinformacije v obstoječo informacijo se nastala informacija ohranja, ker se sicer izgublja oziroma se pojavlja kot t.i. *informacijski šum*. □

*(6)* Cirkularnost in spontanost (nastajalnost) informacije sta *transparentni* za celotno domeno informacije, tj. za informiranje, protiinformacijo, protiinformiranje (informacijsko diferenciranje), vmeščanje (informacijsko integriranje) itd. Cirkularnost informacije je njena *rekurentnost, paralelnost, zaporednost* (posledičnost) itd. □

*(7) Paralelnost/serialnost* informacije je topološko, prostorsko, časovno generativna (nastajalna), spontana cirkularnost. □

*(8) Struktura* informacije so informacijsko sestavljene informacijske oblike in informacijski procesi. *Organizacija* informacije so informacijski odnosi (funkcionalne informacijske povezave) med njenimi oblikami in procesi. Struktura in organizacija informacije sta informacija in oblikujeta t.i. *integracijo* informacije (integrirano informacijo). □

*(9) Informacijski stroj* ima strukturo in organizacijo informacije (npr. informacijo v okviru informacije in stroj v okviru stroja). □

*(10) Informacijski program* informira spontano in cirkularno, nastaja s protiinformiranjem in informacijskim vmeščanjem, se vede kot informacija v okviru informacijskega stroja. Informacijski program more modificirati samega sebe in druge programe in more biti modificiran z drugimi informacijskimi programi med svojim izvajanjem in izvajanjem drugih programov. □

*(11) Metafizika* je označevalec za *celostno* (totalno) informacijo bitja. Metafizične komponente so npr. model sveta, čustvena, senzorska in motorna informacija bitja. Bitje je mogoče razumevati kot *avtopoietski* informacijski sistem. □

### *2.1. Informacija kot sistem v živem sistemu*

Po tem, kar je bilo doslej zapisanega, se informacija pojavlja kot svojski (spontan, cirkularen, nastajalni, prepleteni) sistem. Kaj je v informacijskem sistemu kot pojmovanju sistema bistveno različno od samega pojma informacije? V okviru živega organizma oziroma bitja je informacija sistem procesov, ki se pojavlja v strukturi in organizaciji živega. Vsi ti procesi prinašajo, preoblikujejo, rojevajo in namenjajo informacijo in delujejo sami kot informiranje (procesiranje), od najnižjih molekularnih, celičnih do korteksnih ravnin oziroma od enostavnih do zapletenih informacijskih funkcij. Informacija se kot oblika in proces (kot biokemično pogojno stabilna in spremenljiva struktura in organizacija) pojavlja (razumeva) kot sistem v živem sistemu in tudi kot informacija o samem živem sistemu. Razumevanje informacije kot svobodno dinamičnega procesnega sistema je tako usklajeno z današnjim razvojem zdravega razuma, je kibernetično nastajalno in se ujema z raznovrstnimi znanstvenimi, izkustvenimi in filozofskimi spoznanji in tako ostaja na poti aktualnega spoznavanja.

## 3. Umetni informacijski sistemi

### *3.0. Artificialno kot filozofija, znanost in tehnologija*

Prihodnji človekovi stroji bodo še vedno njegovi artefakti, čeprav bodo imeli strukturo in organizacijo t.i. nevralnih vezij in s tem svojo lastno genetičnost, samoorganiziranost in samorazmnoževalnost. To naj bi bili stroji učenja zlasti na področju njihove uporabe. Vendar bo tudi tem strojem potrebna neka rudimentarna, začetna, izhodiščna struktura in organizacija in v tem segmentu njihovega konstruiranja in nastajanja bo potrebna informacijska filozofija, fundamentalna (strukturna) znanost in nova (biološka) tehnologija. Ti novi artefakti bodo s človekom lahko povezani tudi signalno, preskakujoč človekove senzorje (oko, uho, tip).

### *3.1. Načrt in izvedba informacijskega sistema danes in jutri*

Današnji informacijski sistem je opredeljen kot sistem, *ki je bil razvit za oblikovanje (kreiranje), zbiranje, shranjevanje, procesiranje, razdeljevanje in interpretacijo informacijskih množic* [6]. Današnji informacijski sistem je tako potrebno razviti pred njegovo uporabo. Ta razvoj vključuje analizo potreb, načrtovanje sistema in njegovo realizacijo.

Človeška bitja kreirajo in interpretirajo informacijske mno-žice informacijskega sistema. Informacijski sistem nima vrednosti sam po sebi, če ne olajšuje, pospešuje in izboljšuje človekove aktivnosti. Današnji informacijski sistem tako še nima vgrajene tiste funkcije učenja, ki bi avtomatično zagotavljala njegovo prilagovanje potrebam uporabnika. Ce to sicer velja za človeški faktor sistema pa to prav gotovo ni res pri strojnem delu informacijskega sistema. To pa zahteva nenehno kreativno poseganje (analizo, načrtovanje, realiziranje) človeka v sistem. Jutrišnji informacijski sistem naj bi bil predvsem strojno informacijski tako, da bi interaktivno razvijal svojo strukturo in organizacijo in da ne bi zahteval bistvenega poseganja človeka. To svojo funkcijo samoorganizacije naj bi stroj v sistemu opravljal z rudimentarnimi funkcijami učenja, svoje samostrukturiranje pa naj bi dosegal v okviru tehnologije nevralnih vezij.

### *3.2 Informacijska povezava človek-okolje in človek-stroj*

Informacijska povezava človek-okolje se kaže kot povezava med človekovo metafiziko ter senzorsko in motorno informacijo. Ti trije informacijski pojavi (metafizika, senzorska in motorna informacija) se prepletajo in vzajemno učinkujejo na informacijsko nastajalnost. Simbolično (glej kasneje) lahko to procesnost nakažemo s tremi definicijami:

*nastajanje metafizike* $\mu$ $\quad =_{\mathrm{Df}} \quad \mu,\ \sigma,\ \lambda \models \mu$

*nastajanje senzorske informacije* $\sigma$
$$=_{\mathrm{Df}} \quad \sigma,\ \lambda,\ \mu \models \sigma$$

*nastajanje motorne informacije* $\lambda$
$$=_{\mathrm{Df}} \quad \lambda,\ \mu,\ \sigma \models \lambda$$

$\mu$, $\sigma$ in $\lambda$ oblikujejo človekov paralelni informacijski sistem. Stroj (računalnik) je le posebno okolje, ki podpira človekov informacijski sistem in opravlja zanj specifične podatkovne naloge hitreje, zanesljiveje in algoritmično "objektivno". Povezava človek-stroj je danes vizualna, akustična in taktilna. S pojavom t.i. nevralnih in masivno-paralelnih računalnikov pa bo lahko postala tudi signalna (nevrofiziološka).

## 4. Primeri umetnih, živih in abstraktnih (simbolnih) informacijskih sistemov

### *4.0. Umetni informacijski sistemi*

Umetni informacijski sistemi so tehnološki sistemi upravljanja ali pa so z živimi bitji povezani sistemi, v katerih se interaktivno upošteva tudi živa, bitjevska informacija. Čisti tehnološki sistemi seveda informacijsko razumljeno ne morejo biti pravi informacijski sistemi, saj je pri njih le težko mogoče govoriti o tistem nastajanju informacije, ki je za informacijo glavna značilnost. Ti sistemi so predvsem algoritmični, racionalistični in imajo kvečjemu pogojno predvidljivost, ki pa nikakor ne more nadomeščati informacijske nepredvidljivosti. Takim sistemom lahko rečemo podatkovni sistemi. Mešani sistemi so živi informacijski sistemi s tehnološko podporo in organizacijo; informacijski so prav zaradi žive komponente, zaradi svojih nepredvidljivih akterjev, ki v te sisteme funkcionalno posegajo, jih gradijo in modificirajo, prilagoditveno kreirajo v njihovih življenjskih ciklih. Umetni informacijski sistemi so tako artefakti človeka z njegovo informacijsko udeležbo v njihovi funkciji.

### *4.1. Živi informacijski sistemi*

Živi informacijski sistemi obstajajo na različnih ravneh živobitjevskega. Določene informacijske mehanizme je mogoče opaziti že v t.i. molekulah življenja, kot so DNA, RNA in seveda beljakovine. Te molekule se med drugim značilno pojavljajo v živi celici, kjer oblikujejo celični citoskeletni (mikrotubulni) informacijski sistem [5], od katerega sta odvisni samoorganizacija in samoprodukcija celice. Populacije živih celic sestavljajo organizme, ti pa višje oblike organov vse do človekovih korteksov. V korteksih sta tako struktura kot procesnost informacijski. Nevroni so osrednji procesorji, sinapse pa kompleksni povezovalni procesorji med nevroni. Nevronska mreža se oblikuje po dednostnih zasnovah in tudi pod informacijskimi "pritiski", ki so posledica dogajanja v okolju in interakcije bitja z okoljem. Raznovrstnost je tu očitna: dva nevrona nista enaka, pa tudi ne dve sinapsi in živa nevronska mreža nima dvojnika v svetu živega. Ob vsem tem se trenutni informacijski procesi v nevronski mreži prava značilnost oziroma neponovljiva individualnost dane mreže. V carstvu žive informacije, od celične, organizmične do korteksne, velja ekskluzivni princip pluralnosti, enkratnosti informacijskih individuov in njihove neponovljivosti. Vsak živi oziroma življenjski informacijski sistem je unikat po oblikovnosti (biološki strukturiranosti) in po procesnosti (biološki organiziranosti). Princip informacijske pluralnosti velja tedaj že za molekule življenja (prostorske vijačnice), celice (mikrotubulske vrtiljake) in organizmične podsisteme (nevronske populacije).

### *4.2. Simbolna formalizacija informacijskih sistemov*

Formalizacija informacijskih procesov in sistemov, ki so sestavljeni iz takih procesov, je mogoča z vpeljavo posebne, informacijsko specifične simbolike, in sicer v obliki t.i. informacijske logike [4]. Pokažimo samo formalizacijo z uporabo t.i. splošnih operatorjev informiranja.

*(1) Informacija* $\alpha$ *informira*
$$=_{\mathrm{Df}} \quad \alpha \models \quad \vee \quad \dashv \alpha$$

*Informacija* $\alpha$ *je informirana*
$$=_{\mathrm{Df}} \quad \models \alpha \quad \vee \quad \alpha \dashv \qquad \blacksquare$$

*(2)* Operatorja informiranja $\models$ in $\dashv$ sta splošna operatorja informiranja in ju je mogoče partikularizirati (npr. v znane logične, aritmetične, modalne operatorje) in univerzalizirati na poljubno kompleksen način. Tako imamo npr. tele pomene:

| | | |
|---|---|---|
| $\alpha \models_T$ | pomeni | $\alpha$ *informira pravilno*; |
| $\models_B \beta$ | pomeni | $\beta$ *se verjame;* |
| $\models_B \models_B \alpha$ | pomeni | *verjame se, da se* $\alpha$ *verjame;* |
| $\models_B \alpha \models_B$ | pomeni | *verjame se, da* $\alpha$ *verjame* itd. |

Operatorja $\models$ in $\dashv$ sta lahko unarna, binarna ali multipleksna, infiksna, prefiksna ali postfiksna in predstavljata neomejeni, nastajalni, generativni množici operatorjev (z njuno partikularizacijo in univerzalizacijo). $\blacksquare$

*(3)* Informacija $\alpha$ ne informira

$$=_{\mathrm{Df}} \quad \alpha \not\models \quad \wedge \quad \not\Dashv \alpha$$

Informacija $\alpha$ ni informirana

$$=_{\mathrm{Df}} \quad \not\models \alpha \quad \wedge \quad \alpha \not\Dashv$$ ■

*(4)* S formalnim orodjem, ki smo ga opredelili, je mogoče interpretirati simbolično metaforo na samem začetku spisa. Ta metafora izhaja iz izpeljave tipa modus ponens takole:

$$\frac{\models_{T}(\alpha \models_{K}),\ \models_{T}((\alpha \models_{K}) \Rightarrow (\not\models_{K} \alpha \models_{K})}{\models_{T}(\not\models_{K} \alpha \models_{K})}$$

To pa je prav de Chardinov izrek, kjer je $\alpha$ metafizika (vedenje) živali, $\not\models_{K}$ označuje 'zanesljivo ne vedeti' in $\models_{K}$ 'seveda vedeti'. ■

*(5)* Informacijske operatorje je mogoče partikularizirati in univerzalizirati poljubno na že znan pomen ali na nov, tudi informacijski pomen. Smiselno je mogoče vpeljati 16 specifično informacijskih operatorjev in jih seveda partikularizirati po potrebi, nekatere med njimi pa tudi univerzalizirati. Tako je mogoče uporabljati splošne ($\models$, $\not\models$, $\Dashv$, $\not\Dashv$,), paralelno splošne ($\Vdash$, $\nVdash$, $\dashV$, $\not\dashV$), ciklične ($\vdash$, $\nvdash$, $\dashv$, $\not\dashv$) in paralelno ciklične ($\Vdash$, $\nVdash$, $\dashV$, $\not\dashV$) informacijske operatorje. Na osnovi teh operatorjev je mogoče konstruirati generativni aksiomatični sistem oziroma t.i. informacijsko logiko, ki ponazarja predloženi koncept informacije kot nastajajoče fenomenologije. ■

*(6)* Nazadnje si oglejmo še primer opisa procesa, ki ga v vsakdanjem pomenu imenujemo vpraševanje. Vpraševanje je interaktiven informacijski proces (diskurz), v katerem se pojavljajo zadevna informacija $\alpha$, na katero bo vpraševanje naslovljeno, pojavljajoče (začetno in razvijajoče) vprašanje $q$, in nastajajoči odgovor $a$. Posamezne faze procesa vpraševanja so ciklične (operator $\vdash$), v ciklih pa se pojavlja splošno informiranje (operator $\models$). Pojavitev vprašanja $q$ (njegovo prihajanje v obzorje informacije $\alpha$) je označena z operatorjem začetka ($\lrcorner$), končanje procesa vpraševanja pa z operatorjem konca ($\urcorner$). Primer vpraševanja je tako tale:

$$(\models \alpha) \quad \vdash$$
$$(\models \alpha \Dashv q \lrcorner) \quad \vdash$$
$$((\models \alpha \Dashv q) =_{\mathrm{Df}} (q \models \alpha)) \quad \vdash$$
$$((q \models \alpha \models a) \wedge (a \models q)) \quad \vdash$$
$$((q,\ a,\ \alpha \models q,\ a,\ \alpha) \urcorner)$$

Seveda bi bilo mogoče izraziti tudi bistveno drugačne in bolj zapletene scenarije vpraševanja. ■

## 5. SLOVSTVO

[1] VOLLMER, R., *Alles ist Information,* IBM Enzyklopedie der Informationsverarbeitung, IBM Deutschland, 1984.

[2] DRETSKE, F. I., *Knowledge and the Flow of Information,* Basil Blackwell Publisher, Oxford, 1981.

[3] ŽELEZNIKAR, A.P., *Principles of Information,* Informatica 11 (1987), No. 3, 9-17.

[4] ŽELEZNIKAR, A. P., *Information qua Information,* Zbornik radova Mipro '88, New Generation Computers, 4/15-30.

[5] HAMEROFF, S.R., *Ultimate Computing: Biomolekular Consciousness and Nanotechnology,* Elsevier Science Publishers (1987).

[6] LUNDEBERG, M., G. Goldkuhl, A. Nilsson, *Information System Development,* Prentice-Hall Inc., Englewood Cliffs, NJ (1981).

(V Ljubljani, 22. maja 1988)

# FORUM INFORMATIONIS

## FORUM INFORMATIONIS V ČASOPISU INFORMATICA

V tej številki Informatice objavljamo v rubriki Forum informationis kratko poročilo o zasedanju Forum informationis, ki je obravnavalo problematiko računalniške industrije, nadaljujemo z objavo odgovorov na odprto pismo iz prejšnje številke in naposled prinašamo še primer zares sporne domače strokovne recenzije, ki pomeni bržkone afero par excellence na področju strokovnega in mentorskega dela RiI.

## KRATKO POROČILO Z DRUGEGA ZASEDANJA FORUM INFORMATIONIS

Drugi sestanek FI za okroglo mizo, dne 29. junija 1988, v sejni dvorani SOZD Iskra, v Ljubljani, Trg revolucije 3/XIV, je bil namenjen problematiki računalniške industrije v Sloveniji. Na ta sestanek je bilo vabljenih 50 strokovnjakov, novinarjev, zasebnikov itd. s področja računalništva in informatike (skupaj 28 podjetij in 3 zasebniki). Udeležba je bila s 17 udeleženci dokaj pičla, in sicer: Jože Buh (I Delta), Rado Faleskini (SOZD Iskra), Matija Hudovernik (Radio Ljubljana), Stanislav Klešnik (Intertrade), Marko Kovačević (Mikra), Vinko Kurent (Mikroada, Maribor), Miloš Marinček (FAGG), Marijan Miletić (IJS); Saša Prešern (I Delta in IJS), Mija Repovž (Delo), Slavko Rožič (I Delta), Brane Semolič (Gorenje, SOZD), Peter Stanovnik (Institut za ekonomske raziskave), Maks Svenšek (Birostroj, Maribor), Ivan Šantl (I Delta), Jure Špiler (inovator v svobodnem poklicu) in Anton Železnikar (časopis Informatica).

Vabilo za to zasedanje FI je vsebovalo tale izhodiščna vprašanja: Kaj, kako in koliko proizvajati? Zakaj je lastna računalniška industrija smiselna (potrebna, perspektivna)? Kje so možnosti proizvodnje za izvoz? Kako je z zaščito industrijske in intelektualne lastnine? Kam plove slovenska računalniška industrija? Ali se je že približal trenutek streznitve, ki bi lahko pomenil delitev dela, organizacijo velikih in malih, novo pobudo za zagon industrijske miselnosti?

Oglejmo si na kratko povzetek najmarkatnejših misli, ki so bile izrečene na forumu.

Slovenija nima posebnih političnih in gospodarskih resorjev za področje RiI. Kako naj se v tej praznini razvija računalniška industrija, ki je prepuščena le svoji, večidel nestrokovni spontanosti? Slovenija npr. nima niti ministrstva, niti instituta, niti strokovnega odbora pri GZS za RiI. Celo ponekod v manj razvitih delih Jugoslavije nekaj teh služb imajo. Organizacijska zavest se tako le s težavo oblikuje in ostaja prej ko slej samo koncept kratkoročnega preživljanja. Npr. načrt plasmaja izdelkov domače računalniške industrije na Zapad praktično ne obstaja, oblikuje pa se trenutno še vedno rentabilni plasma na Vzhod. Pri tem je delež tujega znanja v domačih izdelkih ne samo prenizek temveč tudi premalo inovativen, tako da je domače znanje v glavnem že nerelevantno.

V začetku 60-ih let se je v Sloveniji pojavilo podjetje Zuse K. G. ob zastopstvu IBM. Ti pojavi začetkov slovenskega RiI so bili spremljani z intenzivnim šolanjem tudi na univerzi. Že leta 1971 je bila Ljubljana gostitelj svetovnega kongresa IFIP (International Federation for Information Processing). Za to obdobje je bila značilna odločitev, da Slovenija stopa in stopi intenzivneje na pot RiI. Leta 1982 se je pojavil računalniški zakon, ki je nalagal dogovarjanje v razvoju in poslovanju RiI. Zakon je bil posledica občutenega razkoraka med domačim in tujim znanjem, obrambna poteza proti prodirajočim transnacionalkam. Ta zakon je bil posledica samozadostne dobe. Medtem ko se je strategija transnacionalk bistveno spremenila, smo doma pozabljali na nova merila. Kritična zamuda je bila narejena v razdobju med 1970 - 1978. Domače neznanje je postajalo čedalje dražje in neracionalno.

Današnje znanstvene in tehnološke primerjave RiI upoštevajo skoraj izključno le stanje v ZDA in Japonski. T.i. revolucionarni tehnološki skoki se pojavljajo prav v teh dveh deželah zaradi njunega medsebojnega tekmovanja. Vendar ima računalniško industrijo ves razviti svet, tudi Indija. V procesu globalizacije, kjer postajajo veliki prizvajalci še večji, je opazno spajanje producentov in uporabnikov. V tej globalizaciji pa ostaja dovolj prostora tudi za majhne, saj jih vlečejo naprej veliki s sodelovanjem in z vsiljevanjem standardov. Tako se ustvarja nov prostor sožitja velikih in malih - podjetij, narodov in populacij. Slovenija (ali SFRJ) se lahko konkretno primerja s svojimi sosedi: Italijo, Avstrijo in Madžarsko. Računalniška industrija je tudi investicijsko intenzivna in dotok tujega kapitala je nujen. Podjetje Honeywell je npr. že vložilo svoj kapital v proizvodne obrate pri nas.

Računalniška industrija v Sloveniji se sooča s problemi financiranja nabave materiala oziroma s finančno nelikvidnostjo. V bistvu se ta industrija kreditira izključno s prednaročili uporabnikov. Hkrati se pojavljajo (tudi zaradi tega) dodatni problemi s principali (IBM, DEC). Slovenija v razreševanju te problematike že zaostaja, ker ne realizira nujnega prenosa tehnološkega znanja, servisiranja in samostojnejšega razvoja programske opreme. Principali sicer kažejo več razumevanja, vendar določenih pobud ne razumemo ali pa jih ne znamo izpeljati. Za neposrednejše sodelovanje z velikimi računalniškimi podjetji pa mora biti izpolnjena vrsta zahtev, kot so (devizno) izšolani kadri, finančna sredstva za t.i. zagonske investicije, kvalitetna proizvodnja, resnost pri poslovnem dogovarjanju in zanesljivost dobavnih rokov. Skoraj ničesar od naštetega ne obvladamo! Verjetno bi bila najkrajša pot priučitve neposrednejše sodelovanje s tujimi partnerji, začenši z manj zahtevno proizvodnjo in potem naprej s čedalje zahtevnejšo.

V SFRJ obstaja in se še vedno izdeluje strategija tehnološkega razvoja, vendar Slovenija potrebuje tudi svojo, bistveno tehnološko razvojno strategijo, ki mora biti remek delo v primerjavi z jugoslovansko tehnološko globalizacijo. Izoliranost slovenske industrije proti razvitemu svetu je simptomatična: nobeno podjetje ni včlanjeno v ECMA. Brez tega pa resno sodelovanje s partnerji iz evropskega prostora ni mogoče.

Nujna je tista liberalizacija, ki jo poznajo npr. v Južni Koreji, Indiji, Finski in Avstriji; v teh državah je opazen tudi vpliv Vzhoda. In seveda, slovensko podjetje lahko v okviru večje gospodarske liberalizacije uvaža le s svojimi deviznimi sredstvi.

Svetovne poslovne in tehnološke norme niso nikakršnji izumi kulturnih revolucionarjev. Tako je npr. avans posojilo in ne dohodek, inženir ustvarja novo vrednost: daje več kot dobiva! Itd. Veliko tega, kar je svet napravil, je mogoče takoj pobrati. Gorenje ustvarja velik izvoz z velikoserijsko (večmiljonsko) proizvodnjo. Strategija Elana kot uspešnega svetovnega ponudnika je, da je vselej samo drugi (ne prvi!). Računalniška industrija se lahko iz teh domačih primerov marsikaj nauči. Tudi južnokorejske in celo mehiške industrijske serije dosegajo danes stotisoče kosov. V začetku so imeli skupne posle (joint venture) s transnacionalkami, danes pa imajo že lastne zaščitne znamke (trade marks). Proizvodnja, raziskave in razvoj se čedalje bolj internacionalizirajo. Pri zaščiti domačih tržišč so dopustne razlike v ceni le do 15%.

Primer Indije je zanimiv. Indija velja za najbolj demokratično družbo. Takšne družbe imajo izrecne perspektive v svojem razvoju: znanstvenem, tehnološkem in gospodarskem. Indijska demokratičnost je posledica izredno heterogene družbe. Iz tega bi se bilo mogoče kaj naučiti tudi za jugoslovansko demokratično prakso, ki naj ne bi ostajala le izključna razvojna prednost Slovenije. Večja demokratizacija naj bi omogočala tudi organiziranje male proizvodnje, novih institutov in podjetij, v katerih bi neizživeti potencial Jugoslovanov lahko prišel do polne veljave. Tako bi spontano nastajala tudi bolj aktualna kulturna izmenjava demokratične in civilne jugoslovanske misli.

Zasebniki in mala podjetja na področju RiI očitajo delovnim organizacijam (Delti) neodgovornost menaziranja. Ta neodgovornost na različnih upravljavskih ravneh v bistvu odbija sposobne kadre, ki odhajajo drugam in v inozemstvo. Omenjen je bil konkreten primer prof. Egona Zakrajška iz leta 1981; on je danes vodja razvoja podjetja Cromemco v Kaliforniji. Sploh je podobnih spornih kadrovskih primerov kar veliko. Prednost zasebnika je očitna: sam odloča, sam sklepa pogodbe, obrača denar in ima opravka s sodiščem. Želel bi imeti podporo v kaki firmi za butične proizvode. Zasebni inženirji računalništva imajo predvsem manj računalniških orodij in se zato težje znajdejo. Tudi za pridobivanje novega znanja je v malih podjetjih poskrbljeno.

V Jugoslaviji se pojavlja tudi kriza t.i. zastopništva. Jugoslovanski trg stagnira, jug države se sooča z nelikvidnostjo, prodaja je mogoča le še za devizna sredstva pri velikih uporabnikih. Principali ne investirajo, tovarne se gradijo z lastnimi sredstvi le za domači trg. Principali tudi ne dajejo znanja in ne ponujajo možnosti izvoza za izdelke z njihovo tehnologijo! V Avstriji, Italiji in v deželah Beneluksa država podpira uspešne proizvajalce prav pri investicijah (v Sloveniji sta npr. taka primera Intertrade in Delta). Vprašanje o potrebnosti računalniške industrije je najbrž preseženo. Računalniški proizvodi se kot deli in sistemi pojavljajo praktično povsod. Velikoserijska računalniška proizvodnja pa je danes centralno regulirana oziroma omejena z devizno kvoto za material.

A. P. Železnikar

## KRATEK KOMENTAR K DRUGEM ZASEDANJU FORUM INFORMATIONIS

To, kar je na drugem zasedanju FI prav gotovo zbodlo v oči, je bila odsotnost nekaterih glavnih akterjev na področju računalniške industrije v Sloveniji. Zaradi tega je bil diskurz o računalniški industriji morda za nekoga res tako abstrakten, kot je to občutila novinarka Mija Repovž v svojem prispevku "Medla abstraktnost", objavljenem v Delu, 7. julija 1988. Vendar moram glede na gornje poročilo o zasedanju FI prav zaradi tega postaviti vprašanje, za koga je bila lahko vsakdanja problematika računalniške (in druge) industrije, tako kot se je pokazala na FI, le medlo abstraktna. Očitno imamo opravka kar z znatnimi razlikami v razumevanju vsakdanje problematike industrije pri ljudeh v tovarnah in pri tistih, ki informirajo iz salonov naših novinarskih hiš. Verjetno bi lahko novinar uglasil svoje informacijsko ozadje tako, da bi bil sposoben razumevati določeno specifiko industrijske problematike.

Glavna ovira za odkrivanje ključnih problemov domače industrije je še vedno avtocenzura, strah pred akcijo in seveda pomanjkanje določene civilnosti in demokratičnosti v gospodarski organizaciji. Ta demokratičnost se seveda ne nanaša na karkoli, temveč naj bi bila predmet strokovnega in kvalificiranega posluha vodilnih, ki poudarjajo svojo odgovornost. Dokler industrijskih okolij ne bomo civilizirali, tj. odprli za strokovno, organizacijsko, poslovno in predvsem kadrovsko kritiko in za civilne (korektne) odnose med delavci, vse dotlej bo vtečena nesposobnost gospodarila, rovarila in ovirala bistveno učinkovitost (kvaliteto in kvantiteto) proizvodnje.

V Delu, 18. avgusta 1988, se je B. Gruban neutemeljeno spotaknil ob brezbarvno in enostransko razpravo na drugem FI, ki ga je malomarno preimenoval v Forum informaticus. Mislim, da skrajno posplošujoči prispevki v Delu, ki povzdigujejo samozadostnost talentov brez upoštevanja možnosti za npr. industrijsko realizacijo talenta, uveljavljajo predvsem nedoločno namero pisca, ki ni navedel konkretnih ovir in blokad, ki se postavljajo na pot mladim in starim talentom kjerkoli in ne samo na področju RiI. "Znanje za razvoj" v Delu bi na področju RiI velikokrat zaslužilo kar naslov "Neznanje za razvoj".

A. P. Železnikar

## ODPRTO PISMO - NAPOVED ODPRTEGA STROKOVNEGA DIALOGA?

Z dokajšnjim razburjenjem in nekaj prikrite klanske pripadnosti je bilo v različnih krogih, ki se ukvarjajo z informatiko in računalništvom, možno spremljati reakcije na Odprto pismo uredniškemu odboru časopisa Informatica. A priori sem mnenja, da so odprta pisma dobronamerna in zato dobrodošla, zato tudi to potezo šestih bralcev jemljem kot željo in prispevek k spremembi obstoječega stanja na bolje. Verjetno se vsi zavedamo, da je tudi na področju računalništva in informatike obstoječe stanje v družbi nezadovoljivo. Utesnjenost, ki zaradi te nezadovoljivosti nastaja, udari na plano v najrazličnejših oblikah. Poglejmo, kaj so glavne pomanjkljivosti, ki jih na področju informatike in računalništva opažam in nato bomo poskušali ugotoviti povezavo in vlogo časopisa Informatica v tem stanju.

## ZA INFORMATIKO SE SE NISMO ODLOCILI

Čeprav vsak dan govorimo o informacijski družbi, (ponekod že o postinformacijski družbi - econimy of mind), pa trdim, da se naša družba nikoli ni zares odločila za informatiko in računalništvo. Stanje v Sloveniji pa kaže:

- Družbene dejavnosti nimajo sektorjev, ki bi se profesionalno ukvarjali z informatiko:
  * ni Republiškega komiteja za informatiko (imamo pa 10 drugih republiških komitejev)
  * Gospodarska zbornica Slovenije nima Splošnega združenja za informatiko in računalništvo (ima na primer združenje lesarstva, usnjarsko predelovalne industrije, zavarovalstva itd, skupaj 21 združenj).

- Na področju visokega šolstva nimamo Fakultete za informatiko in računalništvo.

- Res je da imamo v uporabi več tisoč računalnikov po delovnih organizacijah, toda ti računalniki niso nikakor orodje informatike. Večinoma niso povezani v integriran informacijski sistem delovne organizacije, ampak so namenjeni raznim skoraj izoliranim finančnim in računovodskim aplikacijam. Delovne organizacije so večinoma še daleč od množične uporabe računalnikov v planiranju, spremljanju in obračunu proizvodnje - to pa lahko prinaša kvalitetna sredstva in kvaliteto dela.

- Računalniška industrija je slabo izkoriščena in nepovezana, za kar je krivih veliko vzrokov. Ponavadi velike računalniške hiše nikoli niso videle svoje vloge v pisanju kvalitetne aplikativne programske opreme in izgradnji integriranih informacijskih sistemov, ampak raje v preprodajanju materialne opreme in zastopniški dejavnosti. Slaba volja, ki jo stresajo na nekaj uspešnih privatnikov ali malih firm, je znak nezrelosti in nesposobnosti velikih organizirati prodorne kadre v skupne projekte.

- V dobi informatike, ko je znanje tržna vrednota, nimamo Instituta za informatiko in računalništvo (imamo 32 inštitutov ali institutov med drugim za biomedicinsko informatiko, za mlekarstvo, za načrtovanje družine, itd.).

- Vrhunski strokovnjaki s področja informatike in računalništva so največkrat tujek v industrijskem okolju. Ambiciozni posamezniki, ki v takem okolju še najdejo voljo do usposabljanja, velikokrat opravljajo podiplomski študij na lastne stroške ali pa celo ilegalno - brez vednosti delovne organizacije, saj računajo, da bi jim magisterij ali doktorat na delovnem mestu utegnil celo škoditi, saj so večkrat priča negativni selekciji.

- Raziskovalni projekti (Evropski, mednarodni) gredo večinoma mimo nas in opažamo tehnološki zaostanek, ki se iz leta v leto veča.

- Industrijsko povezovanje podjetij pri razvoju, proizvodnji in trženju na področju računalniške industrije gre mimo nas, kar nam otežkoča vstop na svetovni trg in nas pušča zadaj v organizaciji in izvedbi masovne proizvodnje računalniške opreme in kvalitetne programske opreme.

## KAJ NAJ CASOPIS INFORMATICA OMOGOCA

V tej precej nezavidljivi situaciji obstaja torej časopis Informatica, ki naj poveže in informira strokovnjake, ki se ukvarjajo z računalništvom in informatiko pri nas. Kaj naj bi bile naloge tega časopisa vemo, strokovnjaki pa so večinoma nezadovoljni s stanjem informatike, čeprav moramo poudariti, da nekateri posamezniki, posebej raziskovalci, dosegajo lepe uspehe.

Časopis naj omogoča, izvaja ali vzpodbuja:

- PUBLICIRANJE STROKOVNIH ČLANKOV: Vemo, da strokovni svet komunicira pretežno v angleškem jeziku in ker naj bi bili prispevki mednarodno usmerjeni, naj bodo napisani v angleškem jeziku. Hkrati naj bi bilo publiciranje v časopisu Informatica priprava za obilico mladih kadrov, da se izvežbajo za pisanje mednarodno kvalitetnih prispevkov (zato tudi v angleškem jeziku).

- INFORMIRANJE: Številne raziskovalne in razvojne aktivnosti na področju informatike in računalništva naj se skupno objavljajo in s tem se omogoči pretok informacij v našem prostoru.

- PROPAGIRANJE: V tako zaostajajoči družbi kot je naša, kjer se tehnološki razkorak med razvitimi in nami tako veča, mislim, da mora časopis Informatica tudi izvesti vlogo propagandne narave, kjer pokaže posamezne pozitivne akcije. Mislim, da je to vlogo časopisa Informatica dobro razumel glavni urednik časopisa.

- KRITIKO: Časopis Informatica se je verjetno premalo ukvarjal s kritiko. Mislim, da lahko k izboljšanju številnih problemov pripomorejo prav posamezniki, ki se profesionalno ukvarjajo z računalništvom.

- POVEZOVANJE: Časopis Informatica naj bi vzpodbujal povezovanje različnih strok, ki se navezujejo na informatiko ali lahko nanjo vplivajo. Tu mislim na področje biologije (genetski inženiring, neuralne mreže), medicine, sociologije, pa tudi filozofije, saj so nekatera vprašanja informatike, posebej paralelnosti procesov in umetne inteligence zelo zanimiva tudi s tega stališča. Na univerzah v tujini (posebej Stanford) so zelo zanimivi mešani interdisciplinarni timi, ki zajemajo informatike, matematike, filozofe, biologe, sociologe in druge ter se ukvarjajo z načelnimi problemi na primer o reprezentaciji znanja v možganih in računalniku, modelih sklepanja in drugo.

## PREDLOGI ZA DODATNE AKTIVNOSTI

Predlagam, da naj časopis Informatica prevzame tudi številne dodatne aktivnosti, ki lahko oživijo in izboljšajo stanje informatike pri nas. Tako naj prevzame:

- IZBOR NAJBOLJSEGA RAZISKOVALCA iz področja računalništva in informatike pri nas (metodologija izbora in kriteriji presegajo cilje tega članka). Mogoče bi ob dobro znanih kriterijih bilo umesno sestaviti rang lestvico vseh raziskovalcev.

- LESTVICO 10 NAJBOLJSIH STUDENTOV računalništva in informatike (to je koristno zaradi pravilnega usmerjanja mladih kadrov, saj vemo, da mnogi potonejo v ubijajočem okolju z negativno selekcijo).

- LESTVICO 5 FINANCNO NAJUSPESNEJSIH DELOVNIH ORGANIZACIJ IN PRIVATNIKOV s področja računalništva (podatke je verjetno mogoče dobiti preko davčne uprave, SDK ali statistike). Kriterij naj bi bil na primer ostanek dohodka na zaposlenega.

- RAZPIS NAGRADNIH RAZISKOVALNIH NALOG ZA STUDENTE - teme naj bodo iz področja računalništva in informatike, zasnovane zelo na široko, tako, da jih lahko izvajajo študenti Fakultete za elektrotehniko, študenti FNT (matematiki, fiziki,...), bilogi, medicinci, filozofi,...

- Objavljanje nekaterih zanimivih in uspešnih raziskovalnih, izobraževalnih ter proizvodnih programov iz tujine. V teh informacijah naj bi bile vključena tudi poročila, kaj so nekatere družbe (države, velike multinacionalke) naredile na organizacijskem, pravnem, finančnem in družbenem področju, da bi pospešile razvoj informatike pri njih.

STROKOVNOST
-----------

Verjetno bi društvo in časopis Informatica pridobil na strokovnem ugledu, če bi poleg dosedanjih aktivnosti organiziral več strokovnih predavanj in zopet gostil kakšen mednarodni simpozij. Mogoče bi bilo zanimivo tudi založništvo strokovne literature.

Korak naprej je tudi akcija, naj časopis dobi ISSN identifikacijo.

Strokovni ugled bi si časopis pridobil tudi z vzpodbujanjem, objavljanjem ali celo izdelavo neodvisnih strokovnih ekspertiz, ki bi zajemale od predloga za celovit nacionalni program informatike do ocene posameznih programskih paketov.

Jasno pa je, da z vlogo, ki jo za časopis Informatica vidim in je po mojem mnenju edino možna v našem okolju, postaja časopis Informatica res informativen in ne časopis vrhunske svetovne kvalitete. Vsakemu bralcu je jasno, da bo svetovno kvalitetne članke šel iskat drugam (če mu bo to dostopno). Takšno je stanje danes.

Saša Prešern

## ODGOVOR NA ODPRTO PISMO

Vsak izobraženec ima določen stil izražanja do katerega ima neizpodbitno pravico, dokler je v skladu z jezikovnimi normami. Ko pa njegov ustvarjalni zagon in svojskost izražanja nista več v mejah jezikovnega standarda, se bralec lahko vpraša, če je tudi vsebina avtorjevega prispevka sploh vredna logične razčlenitve.

Že nagovor so avtorji odprtega pisma končali z dvopičjem, čeprav je pravilno ločilo klicaj. Verjetno so avtorji dobili idejo za uporabo dvopičja v angleškem jeziku, pa še tam se praviloma konča z vejico ali piko, glede na to ali gre za neformalni ali formalni nagovor. Iz slovenskega pravopisa poznamo stvarna lastna imena, v katera sodi tudi "časopis Informatica" in ta se ne sklanjajo, tako da je v rodilniku pravilno "časopisa Informatica" in ne "časopisa Informatice". Lahko se seveda uporablja tudi samo lastno ime Informatica z rodilnikom Informatice. Izgleda, kot da avtorji uporabljajo figure pesniškega jezika, kajti mnogorečje "zaslediti videti" sodi v to figuro, pa še tu je pravilno, da se besedi ločita z vejico. Nekaj stavkov je stilno popolnoma nedopustno napisanih. Tak primer je stavek: Zakaj Informatica objavlja prispevke, ki se prikazujejo kot znanost, pa to niso? Ne glede na to, da je to tudi pesniški način izražanja, kamor sodijo retorična vprašanja, je vsakomur takoj jasno, da se prispevki ne morejo prikazovati, ker gre v takem primeru za metafiziko. Prispevki so v primeru objave v Informatici članki, ki so resda mogoče napisani na magnetnem mediju in v končni fazi natiskani na papir in se prav nič ne prikazujejo. Avtor bi lahko stavek napisal takole: Zakaj so v Informatici objavljeni lažno znanstveni prispevki? Še v tem primeru je taka izjava nekompetentna, ker avtorji očividno nimajo razčiščenih pojmov o tem, kaj je znanost in kaj je lažna znanost, medtem ko celo sami priznavajo, da se na znanstveno fantastiko ne spoznajo.

Najprimernejši sklep tega odgovora na odprto pismo bi bilo znano apelovsko reklo:"Le čevlje sodi naj kopitar!". Ob šestih avtorjih pisma pa sem prepričan, da je imel vsak od njih priložnost, da bi pismo slovnično in smiselno popravil, zato bo tako sestavljeno pismo lahko upravičeno povzročilo bumerangovske odgovore.

Mag. Rihard Piskar, dipl. ing.

## A JOINED DISCOURSE OF SCIENCE FICTION IN COMPUTER SCIENCE (I)

*Anton P. Železnikar*
Iskra Delta, Ljubljana

*That everyone can learn to read will ruin in the long run not only writing, but thinking too.*

Friedrich Nietzsche [-3] 67

### 0. Introduction

*Some of them would want to have the science rented to themselves and albeit they are only mediocre individuals, they behave as a scientific elite; for the sake of appearance they are very active, however, in the essence, they produce merely sterile reports to get or to preserve their position.**

Vinko Kambič [-2] 9

In the open letter [0] to the Journal Informatica, signed by elite Slovene computer-scientists, one of the main reproaches was that some papers published in Informatica belong to science fiction, misrepresented science, and philosophy. Yes, also to philosophy! The last should be one of the most disturbing facts, which are not acceptable by the scientifically

---

* Nekateri hočejo imeti znanost v zakupu in čeravno so mnogokrat samo povprečneži, se tudi vedejo kot znanstvena elita; na videz so zelo aktivni, a v bistvu pišejo le sterilna poročila za dosego ali ohranitev položaja.

(and probably technologically) oriented spirits of foreign scientific journals in the domain of computer science and, of course, to the Scientific Spirit of the open letter signatories. So, let us take an occasional look at some non-scientific quotations, philosophical statements, and ghosts of fiction, taken from eminent computer science journals and books.

For the sake of clarity, we shall put quotations from various sources (e.g. eminent professional journals) being recognized as cata-strophic scientific information (this maybe understood in the sense of the Greek 'kata-strofe' with the meaning 'overthrowing' or 'tricking down') between the particular symbol of catastrophe ϟ⟨number⟩ and the symbol of the end of quotation ⧗ [⟨reference_number⟩]. Beside this, quotations will be printed in Italic. In some cases we shall give short comments on the contents of quotations. Quotations shall not be listed in any systematic order.

## 1. In the Realm of Artificial

*The computer scientist knows, of course. But he certainly does not know exactly what he knows.*

Paraphrasing Teilhard de Chardin

In general, AI (Artificial Intelligence) is a discipline, for which it cannot be said that it belongs to a traditionally solid scientific orientation. In many cases, AI has to do with beliefs, hopes, promises, predictions, goals, programs, methods, and misunderstandings which could be ingenuously classified as non-scientific, fictitious, unbelievable, etc. To step a bit further into pure scientific belief, investigations of this kind should be forbidden, because science has to follow the clearly understood, methodologically legal, and abstractly beaten tracks of research. These tracks of beliefs could probably belong only to the realms of legal mathematics, pure logic, algorithmic computer science, hard-mathematically formalized and semantically reduced cognitive and psychological models, etc. Nowadays, this way of non-thinking and exactly exercising disciplinarity can certainly be scientifically productive, but commonsensically and technologically does not remain on the way of development of modern research and understanding of the living and artificial.

ϟ1 *Almost everyone has an opinion about artificial intelligence (AI). Either it's the wave of the future, or it's just a public relations canard. It's a new programming technology or it's a new mindset - a way of looking at problems that no other discipline has. Or, perhaps the name is an oxymoron and the whole idea of AI is absurd.* ⧗ [1]

Within the domain of AI, we should like to have commonsense reasoning of a living being in a scientific and formalized form to enable an adequate modeling by computational reasoning. However, it is not clear yet how the first kind of reasoning could be projected (modeled) adequately to the second kind of reasoning.

ϟ2 *'We would perhaps be better off using the term "computational reasoning" to represent what we do. Equipped with insights from computer science, we are attempting to look at problems of reasoning and rationality not addressed previously by disciplines other than psychology and philosophy.* ⧗ [1]

AI certainly belongs to disciplines which are not only purely scientific, as are for example philosophy, psychology, cognitive science, neurophysiology, and information processing in living organisms. Wishes and beliefs governing AI can be comprehended merely as goals and speculations of possible research which have not at all entered into the strict scientific, traditionally rationalistic realm yet.

ϟ3 *In AI we use the computational metaphor to help us find answers to questions such as:*
*- How can an intelligent system anticipate the future and prepare for it, without wasting effort on irrelevant or extremely improbable events?*
*- How can we make systems that function with only an impoverished, partial knowledge base in the way that humans are able to do?*
*- How can we communicate successfully with computers in (natural) languages fraught with ambiguity and vagueness?*
*- How can multiple entities plan and cooperate to achieve goals that no individual could ever hope to?*
*While we can look to human behavior for insight into these issues, our main goals are to understand the phenomena of intelligent activity and find ways to exploit that understanding in artifacts.* ⧗ [1]

The questions of the last "cata-strophe" are philosophical for science cannot answer satisfactorily yet. They are maybe on the way of scientific research, but for a pure scientist they may represent an inadmissible approach of exactly structured scientific methodologies. At this point, we can understand how science proceeds on its way of scientific research through philosophy, which opens new horizons and illuminates the way of scientific progressing.

ϟ4 *We are far from being able to answer these questions, or even to integrate what little we know about intelligent behavior.* ⧗ [1]

## 2. Knowledge Representation and Reasoning

*What is knowledge? A traditional answer is that knowledge is a form of justified belief. ... Beliefs can be false, and the truth may not be believed.*

Fred I. Dretske [-1] 85

ϟ5 *A widely recognized goal of artificial intelligence (AI) is the creation of artifacts that can emulate humans in their ability to reason symbolically, as exemplified in typical AI domains such as planning, natural language understanding, diagnosis, and tutoring.*

*Currently most of this work is predicated on a belief that intelligent systems can be constructed from explicit, declarative knowledge bases, which in turn are operated on by general, formal reasoning mechanisms. This fundamental hypothesis of AI means that knowledge representation and reasoning - the study of formal ways of extracting information from symbolically represented knowledge - is of central importance to the field.* ⧗ [2]

To emulate humans in their ability to reason symbolically embraces the entire realm of human metaphysics as information, including the domains such as planning, natural language understanding, diagnosis, tutoring, etc. Saying that most of these activities are based on beliefs accentuates only the metaphysical nature of problems in question. A realistic, good, prospective, and valid science cannot be founded on metaphysical, i.e. individual or personal phenomenology, but has to be constituted by culturally accumulated and disciplined information, which guarantees the existence of a realistic and valid science through selectively extracted information.

↯6 *While the claim that intelligent behavior can arise out of the computational manipulation of propositional symbolic structures is certainly open to argument, AI has not yet developed a serious alternative to this working hypothesis. At the moment, this stance is responsible for the vast majority of work in the field ...* ⧗ [2]

How can intelligence arise out of computational manipulation, if this manipulation does not concern the realm (and nature) of information out of which intelligence can arise as a symbolic structure of information? However, the hypothesis of ↯6 gives some orientation of a possible research when the computational manipulation could be replaced by the informational one. In this case intelligence would arise out of informational manipulation which is understood as Informing of information [10].

↯7 *It should be pointed out that the view of possibility of mechanized intelligence is not new. It seems to have originated with Leibniz' (1646-1716) dream of a calculus of ideas, wherein truths could be determined by manipulating an "alphabet of thought" (characteristica universalis) in some combinatory way, much akin to the way numerical expressions are manipulated in Newton's calculus. The crucial thing about such manipulation is that it would be required to preserve truth, similar to the way that numerical manipulation preserves value.* ⧗ [2]

An alphabet of thought (characteristica universalis), as imagined by Leibniz, can be comprehended in the modern epoch as a kind of logic, say, informational logic [11, 12]. But within such a logic the preservation of truth could only be one of the commonsense aspects, which characteristically belongs to the doctrine of traditional logic. As pointed out in ↯11 (Hintikka), besides the true state of affairs, in the realm of informational, there are a number of other possible states of affairs which could be preserved. The most general principle of preserving could be the principle of information (or principle of cybernetics), irrespective of the informational nature (truth, possibility, temporalness, etc.).

↯8. *Our primary need is for a knowledge-based system to be able to realize all of the things that the sentences in its KB [knowledge base] determine to be true about the world. In other words, when asked a question, $\alpha$, then the system needs to determine whether or not $KB \models \alpha$ ("$\models$" means "entails"). ... Rather, what is taken into account is only the form of the two premises, along with a rule of inference that states that, if it is given $\forall x\ p(x) \Rightarrow q(x)$ ("for any x, if x has property p, then it has property q"), and it is given that $p(a)$ for some particular a ("a has property p"), then it is acceptable to conclude that $q(a)$ holds. We can say that conclusion is derivable from the given premises, and write*

$$\forall x\ p(x) \Rightarrow q(x),\ p(a) \vdash q(a)$$

*(the symbol "$\vdash$" means "derives").* ⧗ [2]

The question is how the sentences in a KB determine the truth about the world. And why only the truth of the world could be relevant? Why not put the question of how information concerning belief, possibility, awareness, necessity, reasoning, understanding or any other informational form or informational process could be preserved? Can really everything be reduced to the problem of truth? Why only truth may sound scientific? Etc.

↯9 *Since proofs are just a matter of form, and not content, it would seem that they are just the kind of symbol manipulation that computers are good at. If we could show this to certain, and could tie the notion of derivation directly to the notion of entailment, then we would have a way for a computer to mechanically produce facts implicit in a KB. Indeed, there are two crucial relationships between entailments and derivations that close the loop for us: soundness and completeness theorems for predicate logic tell us that for a set of sentences, KB,*

$$KB \models \alpha \text{ if and only if } KB \vdash \alpha$$ ⧗ [2]

Deductive proofs are just a matter of form, and not content, and computers may be good at such a symbol manipulation.

↯10 *... the hope was that applications such as weather forecasting, pattern recognition, electronic fault diagnosis and disease identification could all be handled by single software systems.*

*This research was a failure: researchers set themselves too ambitious targets.* ⧗ [4]

The comment to the last "cata-strophe" would be that scientific research can be a failure. So, failures occur within scientific research, irrespective of the strictness of scientific disciplinarity. Certainly, this happens more often in sciences whose principles of doctrinairism are still being developed, i.e. in younger and so-called "life" sciences.

♮11 *The Japanese seizure of expert systems as a medium for their expansion in information technology resulted in a massive explosion in funding. However, in Japan, interest seems to be faltering. There are a number of indications of this.*

*First, good staff are moving from the nationally funded Icot projects back to their companies. Second, few Japanese researchers have published any worthwhile research results: major journals on ai and software engineering are conspicuous by the total absence of any Japanese reports.* ⧗ [4]

It seems that through time institutional funding of research can fail not only in Japan, but also here and now. Have our institutionally organized and funded signatories of the open letter [0] ever asked themselves which way they are moving scientifically, if they have not published any worthwhile research results yet?

♮12 *To build knowledge bases with thousands of rules is a major undertaking, and the resultant expert systems tend to be unreliable and very inefficient. Unfortunately, there has been little funding in this country for establishing the theoretical foundations which might help developers overcome this problem.* ⧗ [4]

Little funding could mean little theoretical foundations for knowledge bases with thousands of rules. Beside this, the present day expert systems are evident products of the so-called artifact builders who do not follow the principles of how the human cognitive mind expertizes problems.

## 3. Belief, Awareness, and Reasoning

Belief, awareness, and reasoning, as rational notions, may belong only to non-scientific or to weak scientific categories. For each of these it is necessary to develop a particular logic, or only one universal logic, which could be adequately particularized for specific cases. The author believes that informational logic [11, 12] can be particularized or universalized to any needed or application-adequate extent.

♮13 *Several new logics for belief and knowledge are introduced and studied, all of which have the property that agents are not logically omniscient. In particular, in these logics, the set of beliefs of an agent does not necessarily contain all valid formulas. Thus, these logics are more suitable than traditional logics for modeling beliefs of humans (or machines) with limited reasoning capabilities.* ⧗ [5]

To be not logically omniscient means to be logically specialized or particularized. Thus, it is possible to construct separate logics for describing beliefs, awarenesses, and reasonings. Each of these logics is characterized by special logical operators, which fit particular notions or semantics. So, we are on the way to search for universal logic, which in particular cases could be induced to the needed particularity.

♮14 *There has long been interest in both philosophy and AI in finding natural semantics for logics of knowledge and belief. The standard approach has been the so-called possible-worlds model. The intuitive idea, which goes back to Hintikka ... , is that besides the true state of affairs, there are a number of other possible states of affairs, or possible worlds. Some of these possible worlds may be indistinguishable from the true world to an agent. An agent is then said to know or believe fact $\varphi$ if $\varphi$ is true in all the worlds he thinks possible.* ⧗ [5]

The last "cata-strophe" stresses the problem of uncertainty, where in the domain of possible worlds not only the true state of affairs is imaginable. Here the philosophy and AI are searching together, trying to find possible semantics for logics concerning different informational phenomena as are knowledge, belief, awareness, reasoning, etc. A strict separation between philosophy as thinking and AI as science is not anymore possible, at least not in the domain of the research of mind which is becoming as philosophical as scientific. In this connection, the injuriousness of an interdisciplinary or common research cannot be at all defended commonsensically.

♮15 *Unfortunately, in real life people are certainly not omniscient. Indeed, possible-worlds advocates have always stressed that this style of semantics assumes an "ideal" rational reasoner, with infinite computational power. But for many applications, one would like a logic that provides a more realistic representation of human reasoning.* ⧗ [5]

It should be said that the protagonists of propositional logic have the intention to make this logic omniscient in the sense of mastering with its reduced means any possible phenomenology. On the other hand, in many applications one would like to have a more adequate or realistic logic, closer to the human way of reasoning.

♮16 *Another interesting direction to take is that of considering quantified versions of these logics. Here some very interesting technical and philosophical questions arise. For example, since we would like to be able to capture sentences such as "He is aware of something that I am not aware of," we seem to be forced into allowing states with different domains, and dealing with all the technical complications that arise there. There is still much work to be done in finding an intuitively motivated logic powerful enough to describe such situations.* ⧗ [5]

Article [5] begins with the quotation of Teilhard de Chardin: *"The animal knows, of course. But it certainly does not know that it knows."* This quotation is similar to that "He is aware of something that I am not aware of", from ♮13. In [6], the author described de Chardin's statement by the formula $\not\models_k \alpha \models_k$. In this formula $\alpha$ is the the so-called animal's metaphysics (an animal's total information), $\not\models_k$ marks "certainly does not know" and $\models_k$ denotes "knows". In general, the meaning of this formula would be that information $\alpha$ is not informed that it informs. In the second case we can put the following equivalences:

$$\Upsilon =_{Df} (\text{'he'} \equiv \text{'his metaphysics'})$$
$$\omega =_{Df} \text{'something'}$$
$$\alpha =_{Df} (\text{'I'} \equiv \text{'my own metaphysics'})$$
$$\models_a =_{Df} \text{'is aware'},$$
$$\not\models_a =_{Df} \text{'is not aware'}$$

Now the formula which describes the second case is, for example,

$$\alpha \not\models_a \omega \models_a \Upsilon$$

Already in [7] several cases of this kind of statements are formally described:

$\models \models \alpha$: $\alpha$ is informed that it is informed;
$\not\models \models \alpha$: $\alpha$ is not informed that it is informed;
$\models \not\models \alpha$: $\alpha$ is informed that it is not informed;
$\not\models \not\models \alpha$: $\alpha$ is not informed that it is not informed;
$\models \alpha \models$: $\alpha$ is informed that it informs;
$\not\models \alpha \models$: $\alpha$ is not informed that it informs;
$\models \alpha \not\models$: $\alpha$ is informed that it does not inform;
$\not\models \alpha \not\models$: $\alpha$ is not informed that it does not inform;
$\alpha \models \models$: $\alpha$ informs that it informs;
$\alpha \not\models \models$: $\alpha$ does not inform that it informs;
$\alpha \models \not\models$: $\alpha$ informs that it does not inform;
$\alpha \not\models \not\models$: $\alpha$ does not inform that it does not inform.

In these formulae, the particularization (substitution) of the operators $\models$ and $\not\models$ can be non-uniform. For the first formula, for instance, $\models_T \models_B \alpha$ has the meaning, $\alpha$ is informed true that $\alpha$ is believed, etc.

♮17 *Two remarkable trends in automation can be seen. One is "intellectualization" and the other is "systematization". The former aims to improve the performance of machines to the level of a human operator.* ⧖ [8]

To improve the performance of a today's machine to the level of a human operator may sound unrealistic or at least non-scientific. However, the intention of scientific research has to remain in the domain of unknown, otherwise it cannot follow the aim of any research: to reveal the unrevealed.

♮18 *The computer is superior to the human brain in memory, information retrieval and logical reasoning, but inferior in intuition, macroscopic judgment, commonsense knowledge penetration, creation, adaptation, flexible thinking and so on. In a word, the computer can perform the part of the left hemisphere of brain but not the right. Therefore, we should reconsider the role of the computer as a part of the man-machine system..* ⧖ [8]

That computer can perform the part of the left hemisphere of brain but not the right may sound as a psychological paraphrase but not as a legal scientific statement. Besides, this statement is very ambitious even if it is said that computer can perform only a part (what functions of the part) of the left hemisphere activities. Yes, this could be a computer from a science fiction story. But, it is certainly also true too that many performances of computers are comparable to the functions of human mind.

♮19. . . .*special commonsense knowledge is a meta-rule which covers the deficit, harmonizes the contradiction, and increases the adaptability and flexibility of the special knowledge.* ⧖ [8]

Some readers would agree that the last "cata-strophe" is philosophical and can hardly be brought into a legal scientific context.

## 4. Philosophy Concerning Computer Science

♮20 *Epistemological issues have been a concern of Western philosophy at least as far back as the Sophists ... . Cognitive science, on the other hand, is still in its childhood as an interdisciplinary pursuit which attempts to integrate the resources of artificial intelligence, cognitive psychology, philosophy, and perhaps other areas. ... it is now time for epistemology to begin to reap the benefits of cognitive science.* ⧖ [9]

Cognitive science, as described in the last "cata-strophe", as a field of interdisciplinary research essentially concerns AI and through it the entire domain of the computer science. Psychology and philosophy influence the research in AI as well as in computer science quite substantially. Due to specific requirements of the AI field, especially particularized philosophies evolve, which may speed up the development of a new scientific and technological understanding.

♮21 *Further, Stefik and Bobrow belong to a predominant tradition within AI that sees little value in foundational (called "philosophical") questions. Shortcomings of current techniques and systems are seen as a transient failure to cover the right details and to invent the right clever mechanisms, rather than as anything more basic. Instead of radical challenge to foundations, they would rather see arguments over the technical details ... They imply that if we focus on the details of the technology, somehow it will all work out. They indulge in science-fiction speculations, such as "... never computer systems ... might acquire background that we humans bring to bear to understands the world classes of complexity for computational beings that relate in natural ways to Piaget's stages novels of Isaac Asimov. This kind of innocent optimism about technological achievement has been common among AI researchers since the beginning of the discipline.* ⧖ [13]

Obviously, the signatories of the open letter [0] also belong to a predominant and additionally subcultural tradition within the domestic computer science, that finds no value in philosophical questions by which foundations could be widened. They cannot see anything more basic, but they believe, that the present day shortcomings in computer science are a transient failure, which will vanish through time when traditional means will be developed in those details which now imply failures. And right this kind of concluding let them speculate in the domain of science fiction by the same sentences as quoted in the last "catastrophe".

## 5. Of AI Scientists

Computer scientists are becoming more and more AI scientists. They are the creators and explorers of that what we may call computer science and artificial intelligence. Thus, they have their own way of understanding what the science in which they create may illuminate, enlighten, or bring to the surface of their own consciousness. But, there is a substantial difference in thinking between computer scientists and AI scientists: the first are more like hard-workers, dwelling in a computer-specific hard and soft doctrines; they walk through a computer land like somnambulists; the second are like discoverers, breeding soft and disciplinary open doctrines. But, there are also substantial differences in the informational environments of the first and the second ones.

♮22 *There are two types of AI scientists; call them P and not-P. Here is a simple test to tell which type you are. You are type not-P if you immediately close a book containing sentences like this: "We are led, according to Descartes, to an agent that circumvents the hegemony of the mechanical-deterministic nexus by virtue of its immateriality and freedom ..." If you are type not-P, skip this book; you will not miss any important new clues about writing intelligent programs. Type P readers, however, will find it a useful and at times illuminating discussion of mayor concepts in the philosophical and psychological analysis of mind.* ⧗ [3]

The signatories of the open letter [0] belong obviously to the not-P scientists. The advantage of P scientists is that they can read and find useful to get insight into the philosophical and psychological analysis of mind. If nothing else, the not-P scientists remain narrowed in their "fach" doctrines, and utilitarian in respect to any philosophical enlightenment which might come from new ideas, concepts, methodologies, etc. The not-P scientists are the real guardians of their scientific disciplines, keeping the purity and unchangeability of their disciplines under the severe control. It is evident that in scientific research the not-P scientist will in principle not support a new, innovative research achievement.

♮23 *As it is, it is still one of the best of the recent attempts to find, explain or construct the appropriate philosophical and psychological context for AI. As such it is one of the few books available on the mythical subject of cognitive science.* ⧗ [3]

No human information is completely free of mythical beliefs. Not only the cognitive science, but also AI and computer science include myths in believing that intelligent objects can be constructed through non-intelligent mechanistic items. But how can a hard, scientifically declared discipline be other than mythical, if one of the most relevant goals is the preservation of unreal beliefs and of solid professional organization (organicism)?

♮24 *The rise of neural computing can be seen as part of the interplay between two schools of thought in ai: the cognitists and the artifact builders.*

*The artifact builders are interested in building software systems which exhibit intelligent behaviour and do not concern themselves with whether the underlying mechanisms of the software resemble the way humans operate or not. The cognitists are interested in using the computer as a laboratory for exploring notions about the way human think.* ⧗ [4]

In some way, the not-P and P scientists can be compared to artifact builders and cognitists, respectively. There exists some congeniality between a not-P scientist and an artifact builder. The first one builds his science irrespective of a relevant philosophy and the second one constructs his artifacts as useful tools or even as plausible toys. On the other hand, a P scientist and a cognitist is very sensible to philosophy (e.g. Descartes, Pascal, Leibniz, Hilbert, Russel, Wittgenstein, etc.), so he can innovate the existing science by bringing a new semantics into the scientific context.

## 6. Some Comments to the First Part

The intention of listed quotations and comments to them in this part of the discourse was to show how some listed quotations can be understood and misunderstood as science, science fiction, or misinterpreted science. This is the way of information development in the field we call scientific research. In the area of research, beliefs, awarenesses, and reasonings follow wishes, hopes, and cares for the future, survival, but also for individual scientific progress, individual curiosity, and certainly the will for power. Scientific world is in no way free from personal aspirations, narrow interests, and ideologies of knowledge. In the history of science this kind of dilemma appears as a regular one. "Every triumphant theory passes through three stages: first it is dismissed as untrue ..." (embryologist Von Bear). In this context, the signatories of the open letter [0] may appear as the guardians of what has been said at the beginning of Introduction: some of them would want to have the science rented to themselves and ...

## References

[-3] F. Nietzsche: *Thus Spoke Zarathustra*. Penguin Books (Reprinted 1985).

[-2] V. Kambič: in *Knowledge for Development*

(in Slovene). Delo, Aug 11, 1988, Ljubljana.

[-1] F.I. Dretske: *Knowledge and the Flow of Information.* Basil Blackwell Publisher, Oxford (1981).

[0] T. Pisanski & All: *An Open Letter to the Editorial Board of the Journal Informatica* (in Slovene). Informatica 12 (1988), No. 3, 78-79.

[1] R.J. Brachman, F.H. Henig: *The Emergence of Artificial Intelligence Technology.* AT&T Technical Journal 67 (1988), Issue 1, 3-6.

[2] R.J. Brachman: *The Basics of Knowledge Representation and Reasoning.* AT&T Technical Journal 67 (1988), Issue 1, 7-24.

[3] O.J. Flanagan: *The Science of Mind.* Book Reviews and Response (Reviewed by R.K. Lindsay). Artificial Intelligence 34 (1988), 385.

[4] D. Ince: *A New Dawn Breaks in Japan as Research Sinks in the West.* Computing (March 17, 1988) 18-19.

[5] R. Fagin, J.Y. Halpern: *Belief, Awareness, and Limited Reasoning.* Artificial Intelligence 34 (1988) 39-76.

[6] A.P. Železnikar: *Information and Information System* (in Slovene). Proceedings of the Conference on Information Systems, 7-16, Technical Faculty, Maribor, June 22-24, 1988.

[7] A.P. Železnikar: *Information qua Information.* Proceedings of MIPRO '88, New Generation Computers, 4/15-30.

[8] T. Terano: *Fuzzy Logic in Man-Machine Systems.* New Generation Computing 5 (1987) 131-132.

[9] S.W. Smoliar: A Review of *A.I. Goldman, Epistemology and Cognition.* Artificial Intelligence 34 (1988) 251-267.

[10] A.P. Železnikar: *Principles of Information.* Cybernetica 31 (1988) 99-122.

[11] A.P. Železnikar: *Informational Logic I.* Informatica 12 (1988), No. 3, 26-38.

[12] A.P. Železnikar: *Informational Logic II.* Informatica 12 (1988), No. 4,3-20.

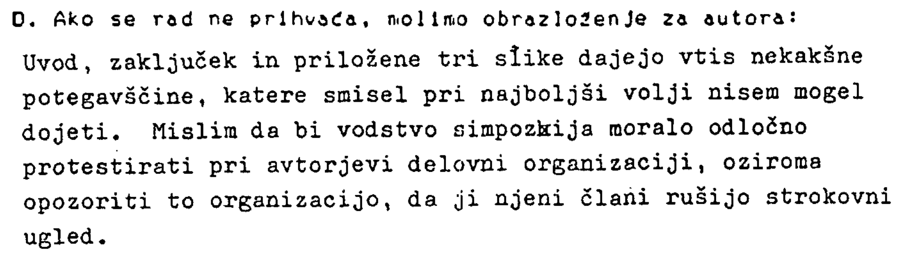
[13] T. Winograd, F. Flores: *On Understanding Computers and Cognition: A New Foundation for Design.* A response to the reviews. Artificial Intelligence 31 (1987) 250-261.

## O RECENZIJAH

Razumljivo je, da so recenzije o istem delu lahko nekoliko različne, pač odvisno od recenzentovega natančnega vpogleda v področje, ki ga recenzira, od njegove osebnosti, pa tudi od njegovega trenutnega razpoloženja. Težje pa razumemo bistveno različnost recenzij istega dela. Kot primer bi nanizal recenzijo prispevka, ki je bil poslan na jugoslovanski mednarodni simpozij (priloga 1) in odgovor urednika ameriške revije (priloga 2), kamor je bil poslan isti prispevek v prevedeni obliki. Sodbo o enakosti vsebin recenzije in odgovora si ob branju ustvarite sami.

(I.R.)

(priloga 1)


D. Ako se rad ne prihvaća, molimo obrazloženje za autora:

Uvod, zaključek in priložene tri slike dajejo vtis nekakšne potegavščine, katere smisel pri najboljši volji nisem mogel dojeti. Mislim da bi vodstvo simpozkija moralo odločno protestirati pri avtorjevi delovni organizaciji, oziroma opozoriti to organizacijo, da ji njeni člani rušijo strokovni ugled.

(priloga 2)

I believe that you have an interesting *idea* for an article here, but I would like to see more details of the language and more description of what it is used for (is it for Computer-Aided Design, Computer-Aided Manufacturing, or what?) There are questions about some of the things contained in it that might be cleared up by a more extensive exposition. For example, you state "single call rule: every subroutine can be called once only." If that were **strictly** true, it would be more economical to code the subroutines in-line; what I assume you mean is that subroutines are not re-entrant — there can be at most one invocation of the subroutine at any given time.

The artwork is interesting, and I would be happy to publish it together with an expanded article, IF you could bear in mind that the standard format on this side of the ocean is 8.5" x 11" (28 cm x 21.5 cm). Usually the foreign submissions that I receive have margins that can be trimmed; that is not the case on the drawings.

If you care to resubmit the article, or any others, I will be happy to consider them. I don't know whether you are a member of SIGSMALL/PC or not; if not, and you don't get the newsletter, I will say that I have rewritten your previous contributions for the sake of grammar and spelling; I hope that this causes you no distress.

[illegible]
H. K. Hodge
Editor, SIGSMALL/PC Notes

A SHORT COMMENT TO ENGLISH READERS

The upper Enclosure 1 (priloga 1) is a Slovene referee's report to the program committee of a Yugoslav computer conference. This Enclosure has to be compared to Enclosure 2 (priloga 2) which is the editor's of SIGMALLS/PC Notes opinion on the same paper. The English translation of Enclosure 1 is the following:

D. If the paper is not accepted we would like to have an explanation with arguments for the author:

***The introduction, conclusion, and the three drawings enclosed give the impression of a kind of fraud, which to my best will, I could not comprehend. I think that the committee of the symposium must resolutely express a protest at the author's working organization or let this organization know that its employees destroy its professional reputation.***

(The translator regrets the inconvenient diction of the translation of the Slovene original into English which is typical for the herostratic style of the referee's opinion. By this diction the spirits of the referee's opinion remains true.) APŽ

## ŠE O RECENZIJAH

A. P. Železnikar, Iskra Delta

Ker gre v prispevku "O recenzijah", v katerem je priloženo tudi dokazno gradivo, za značilno domačo afero, naj mi bo dovoljeno, da dodam svoje skromno mnenje kar v domačem jeziku. Takoj na začetku ponavljam vprašanja o histeričnem pamfletiranju, rotenju, zaklinjanju, herostratstvu oziroma o motivativnih konfliktih starejše generacije, ki sem jih načel v [1]. Kako se obraniti teh strastnih in čustvenih (in mestoma že kar sovražnih) izlivov in njihove škodljivosti pri prodiranju novih akterjev oziroma njihovih izvirnih dosežkov na domačih prizoriščih v okviru izrazite znanstvene subkulturnosti? Ali bomo morali začeti resneje in konkretneje postavljati vprašanja, kdo so ti naši, nam neznani avtoritativni usmerjevalci v znanosti, ki povzročajo zaradi svoje preživele strokovnosti blokade tam, kjer jih domala nihče več ne pričakuje. Kot da smo priča nekemu razvojnemu rasizmu, ki nam omejuje in predpisuje spontan in s svetom usklajen razvoj domače znanstvene misli. Seveda nikakor ne trdim, da so pojavi recenzentskega rasizma prisotni samo na področju računalništva in informatike, zasledimo jih lahko tudi pri fizikih, zlasti v segmentih intenzivnega in filozofsko pogojenega znanstvenega razvoja.

Upam seveda, da gre v primeru navedene recenzije res za osamljen pojav, katerega nosilec in izvajalec ima v širšem jugoslovanskem okolju le obrobni (slovensko značilni, subkulturni) vpliv. To pa seveda ne odvezuje odgovornosti organizatorjev strokovnih srečanj, za katere se takšne recenzije pišejo. Ker, kakšni pa bodo postali domači simpoziji in posvetovanja, če bo njihova strokovna vsebina odvisna od ljudi, ki nastopajo kot čuvarji balkanskega kodeksa? Njim bi veljal preprosti nasvet, da v svoje recenzentske garniture ne vključujejo kogarkoli, zlasti pa ne privržencev herostratstva, ki bi bili sposobni zadušiti redke pojave domače znanstvene in tehnološke ustvarjalnosti.

Seveda bi bila tudi moja reakcija na omenjeno recenzijo zgolj čustvena, če ne bi znal argumentirati in po svoji vesti interpretirati vsebine žaljive recenzije. Organizator posvetovanja pričakuje pod D korektno strokovno obrazložitev zavrnitve, ki bo sporočena avtorju. Pričujoča zavrnitev pa ni le napad na integriteto avtorja, temveč je še poziv na sankcijo proti ugledu njegove delovne organizacije. To pa je hkrati argument, ki ga lahko izkoristi nekdo drug, recenzentskemu herostratu podoben herostrat, npr. pri visokošolski oziroma strokovni reelekciji avtorja. Strokovne utemeljitve zavrnitve prispevka v recenziji seveda ni, je pa motivativni konflikt, ki se izraža v izjavi, da je prispevek informacijsko tako oddaljen od informacijskega ozadja recenzenta, da ga ta ne more več dojemati pri vsej svoji volji za razumevanje prispevka. Prav tu pa nastane vprašanje strokovne usposobljenosti recenzenta in njegove normalnosti: kakšno je njegovo informacijsko strokovno ozadje, da prispevka ne more več dojemati in ga prav zaradi te svoje nesposobnosti imenuje potegavščina. Očitno tak recenzent nima več kaj iskati v strokovnem informacijskem prostoru, v katerem deluje ne le kot recenzent temveč kot akter stroke, raziskovalec, znanstvenik, pedagog, inženir in naposled tudi kot človek.

Prepričan sem, da je gornja osvetlitev vsebine recenzije zadostna; sedaj pa bi bilo mogoče še povedati, kaj bi v negativni recenziji pričakoval kot pojasnilo za avtorja. Argument za zavrnitev bi lahko bil: neoriginalnost, premajhna izvirnost, neupoštevanje standardov stroke, konkretni deli prispevka, ki so utemeljeno nesprejemljivi, napačni rezultati, nerazumljiv jezik, konkretna poprava in dodelava in kar je še podobnih trdnih kriterijev. Veliko tega pa je pokazanega prav v prilogi 2, kjer si je avtorju popolnoma tuj urednik vzel prizadevno toliko časa, da mu svetuje in pomaga prek oblikovnih in vsebinskih težav. V tem se razpoznava konstruktivni mentorski odnos in odgovornost recenzenta do avtorja. Seveda se recenzent lahko postavi tudi na stališče, da ocenjuje posplošujoče takole: znanstvena fantastika, lažiznanost, čudne izjave, s stališča recenzenta nerazumljivo, sramotno, skrb zbujajoče, strokovno rušilno, neznanstveno itd. Toda prav to se od strokovnega recenzenta ne pričakuje. Pomislite, kako bi lahko razsojalo sodišče, ki bi uporabljalo posplošujoče kriterije na nedoločen in voluntarističen način. V tem primeru stroka ne bi bila kaj več od političnega voluntarizma.

Nazadnje bi seveda lahko postavil še vprašanje prizivnosti avtorja in njegove delovne organizacije. Avtor je del svojega priziva prav gotovo izrazil z objavo dokumentov. Vendar je tu zasluga za njegovo pozitivno obrambo zgolj njegova in se lahko zaradi njegove lastne iznajdljivosti dobro konča tudi zanj. Mislim pa, da bi morala avtorjeva delovna organizacija izraziti tehten pomislek jugoslovanskim organizatorjem posvetovanja, ki so spregledali nestrokovnost in (nezavedno) zlonamernost recenzenta, predvsem pa nedopustno mešanje strokovnih in zasebnih (konfliktnih) podmen. Ali to pomeni, da balkanski strokovni simpoziji blokirajo informacijo prav tam, kjer je ta mednarodno relevantna? Če recenzenti ne morejo več pri svoji najboljši volji razumevati razvojnih tokov "svoje" stroke po svetu, potem pač učinkovito širijo svojo strokovno nesposobnost povsem nočeš-hočeš na perspektivne domače avtorje.

## Slovstvo

[1] A.P. Železnikar: Še o soncih v praznem prostoru. Informatica 12 (1988), št. 3, 81-83.